


Week 10: *Polishing Charts*

 EMSE 4572: Exploratory Data Analysis

 John Paul Helveston

 November 02, 2022

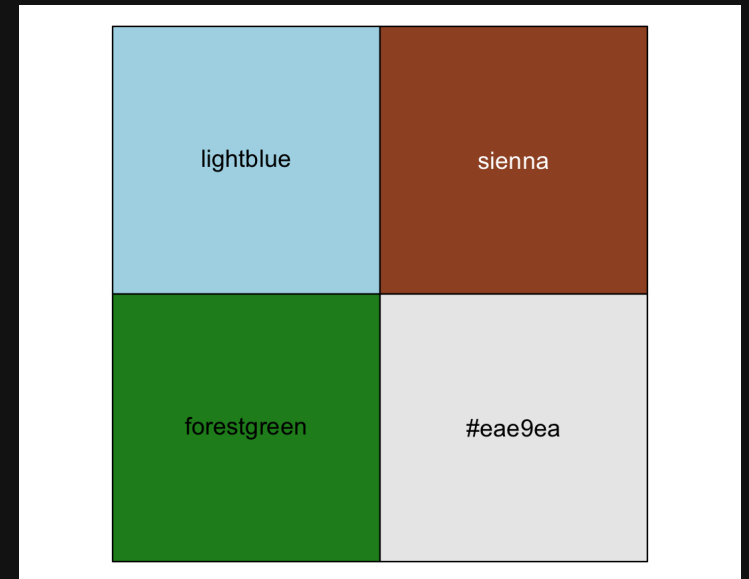
Tip of the week

Ever wondered what colors look like *before* you plot them?

Use `scales::show_col()` to preview them

Example:

```
colors <- c('lightblue', 'sienna', 'forestgreen', '#eae9ea')
scales::show_col(colors)
```



Today's data

```
wildlife_impacts <- read_csv(here::here('data', 'wildlife_impacts.csv'))  
federal_spending <- read_csv(here::here('data', 'federal_spending_long.csv'))  
milk_production <- read_csv(here::here('data', 'milk_production.csv'))  
lotr_words <- read_csv(here::here('data', 'lotr_words.csv'))  
msleep <- read_csv(here::here('data', 'msleep.csv'))
```

New (?) package:

```
install.packages('hrbrthemes')
```

Week 10: *Polishing Charts*

1. Scales

2. Annotations

BREAK

3. Colors

4. Fonts

5. Rmd tricks

Week 10: *Polishing Charts*

1. Scales

2. Annotations

BREAK

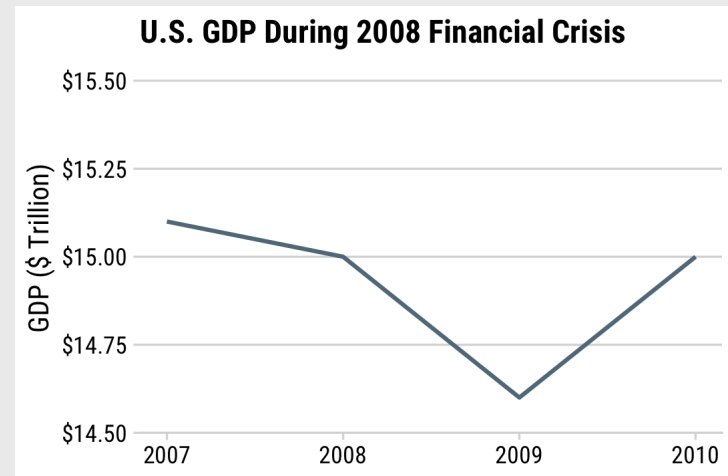
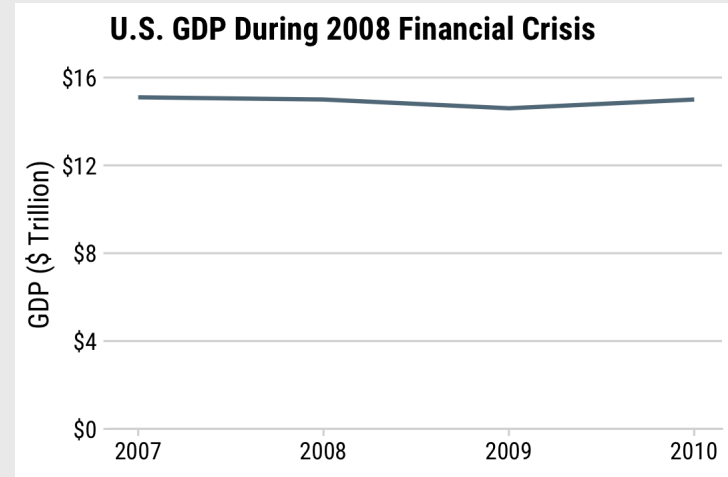
3. Colors

4. Fonts

5. Rmd tricks

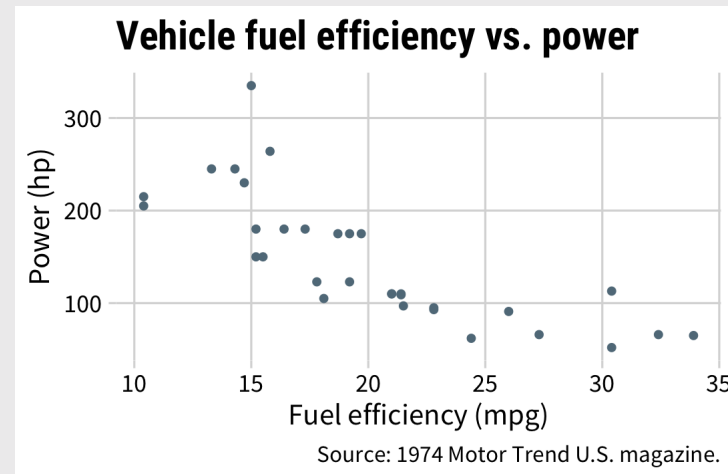
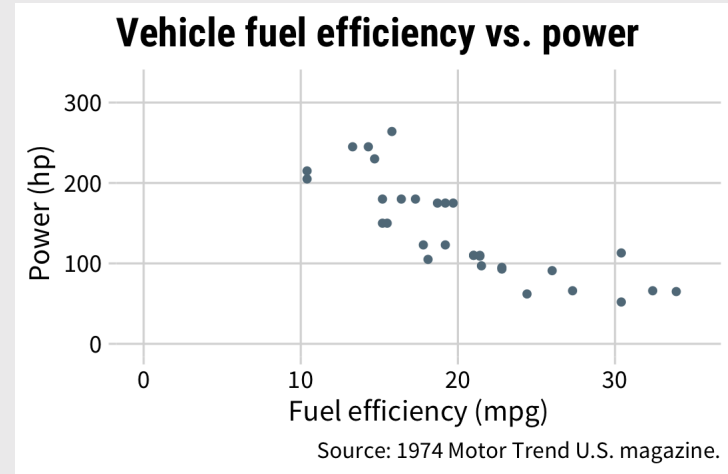
When is it okay to truncate an axis?

- **When small movements matter**



When is it okay to to truncate an axis?

- When small movements matter
- **When zero values are impossible**



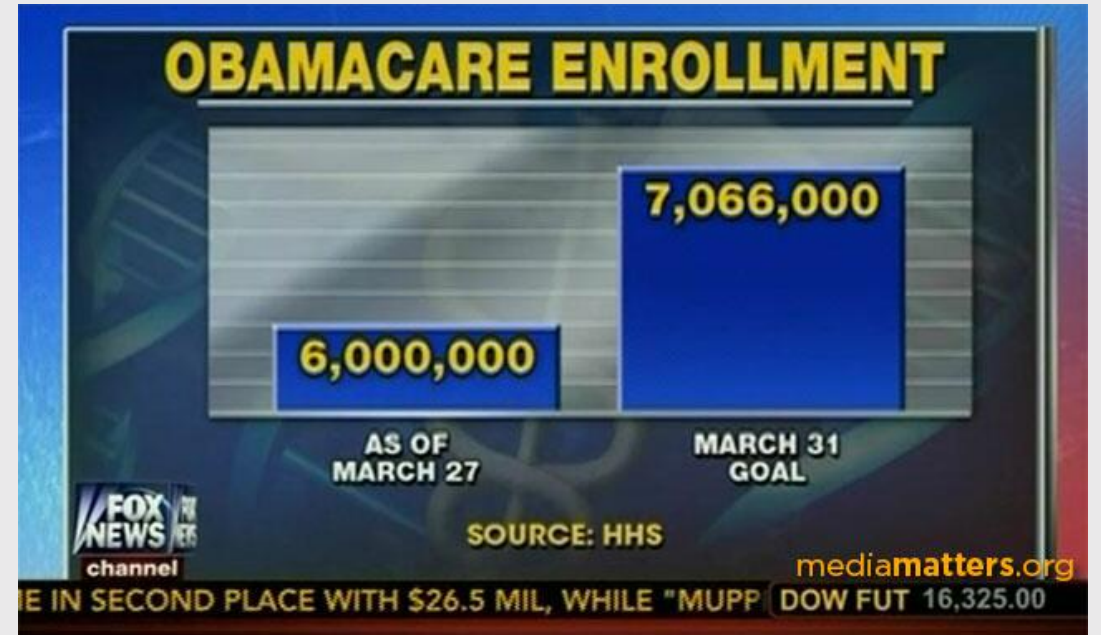
When is it okay to truncate an axis?

- When small movements matter
- When zero values are impossible
- **When it's normal / a convention**



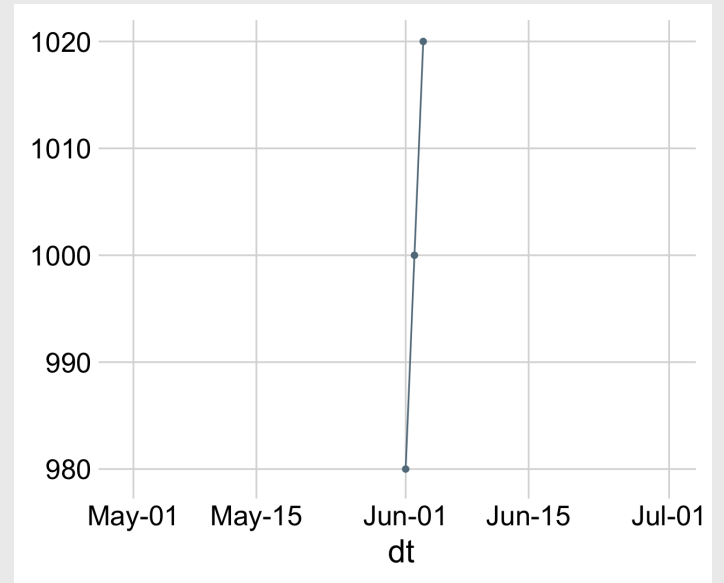
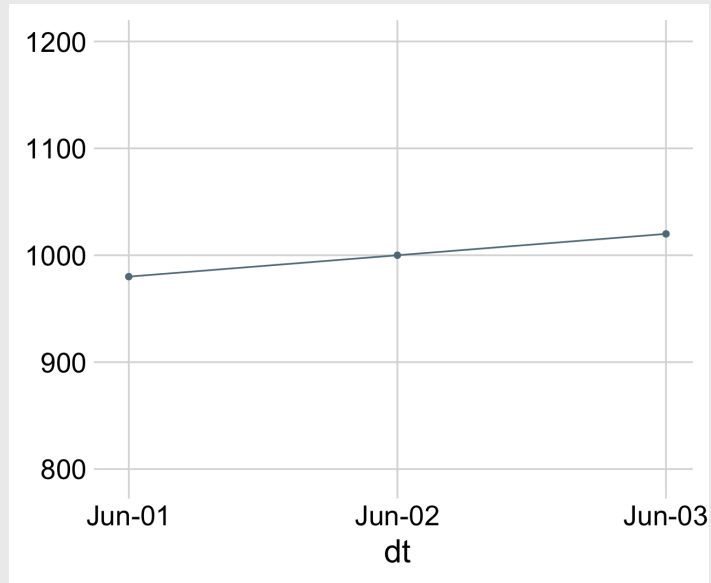
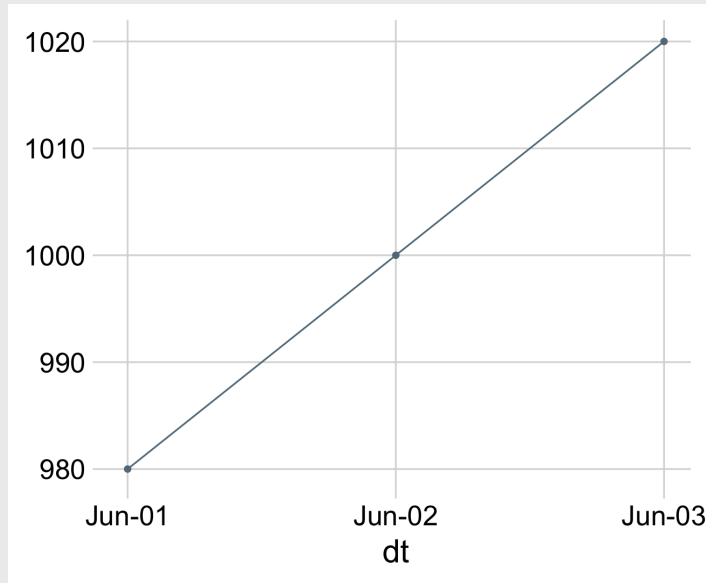
When is it okay to to truncate an axis?

- When small movements matter
- When zero values are impossible
- When it's normal / a convention
- **Never on a bar chart**



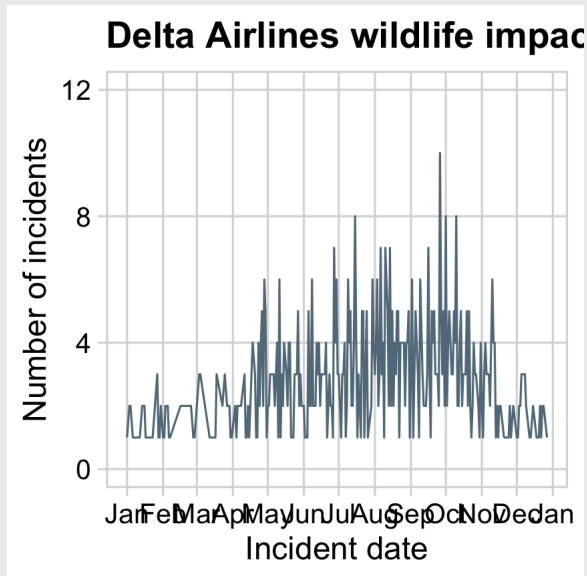
You are most sensitive to changes
in angles close to 45 degrees

You are most sensitive to changes in angles close to 45 degrees

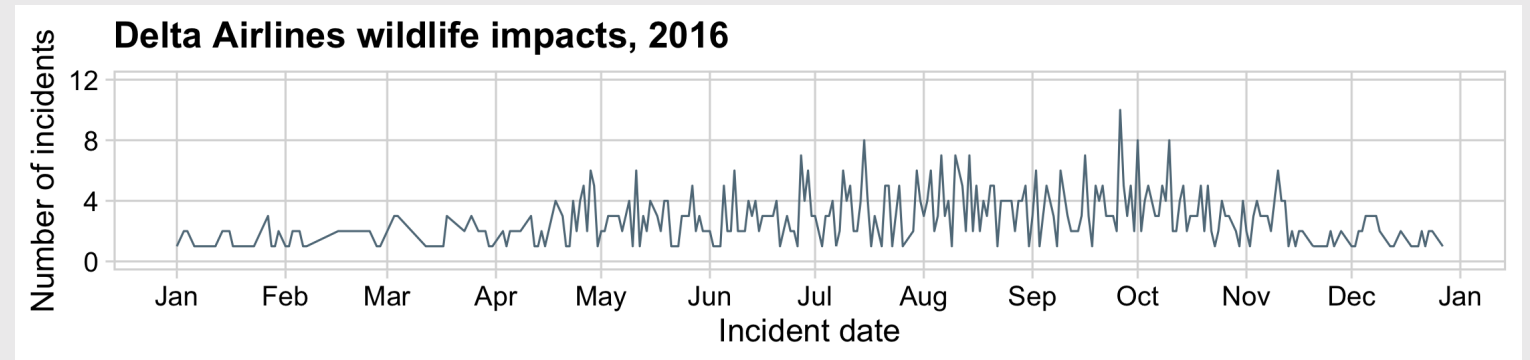


You are most sensitive to changes in angles close to 45 degrees

Bad



Better

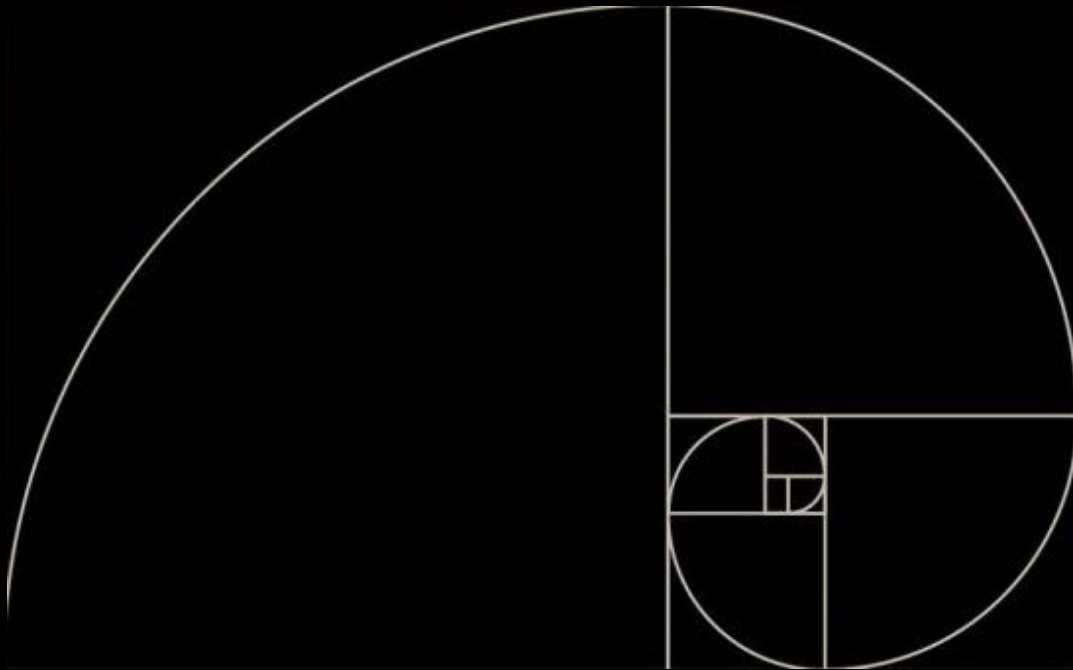


Set image dimensions in R chunk header:

```
```{r, fig.width = 5, fig.height = 3.1}
```



Consider setting dimensions to "Golden Ratio" (1 : 1.618)



Approx. to golden ratio:

<b>width</b>	<b>height</b>
5	3.1 or 3
6	3.7 or 4
7	4.3

[Also check out Donald Duck in Mathemagic Land](#)

# Adjust axes with `scale_*` functions

## Continuous variables

```
scale_x_continuous()
scale_y_continuous()
```

## Discrete variables

```
scale_x_discrete()
scale_y_discrete()
```

## Others

```
scale_x_log10()
scale_y_log10()
scale_x_date()
```

## Common arguments for **continuous** variables

```
scale_y_continuous(
 # Set the lower & upper boundaries
 limits = c(lower, upper),

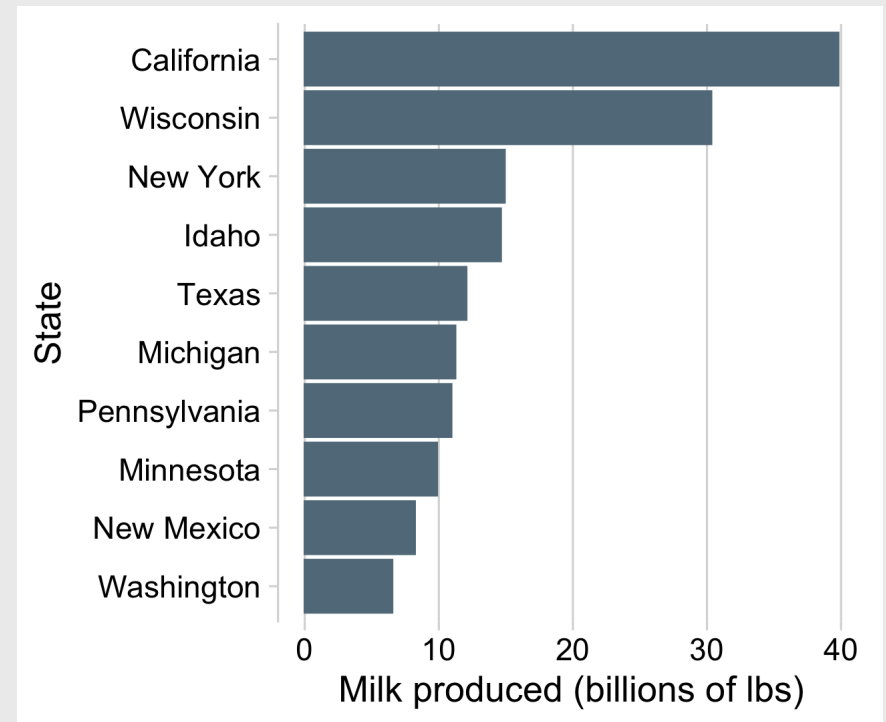
 # Explicitly set the break points
 breaks = c(break1, break2, etc.)

 # Adjust the axis so bars start at 0
 expand = expand_scale(mult = c(0, 0.05))
)
```

# Adjusting **continuous** scales

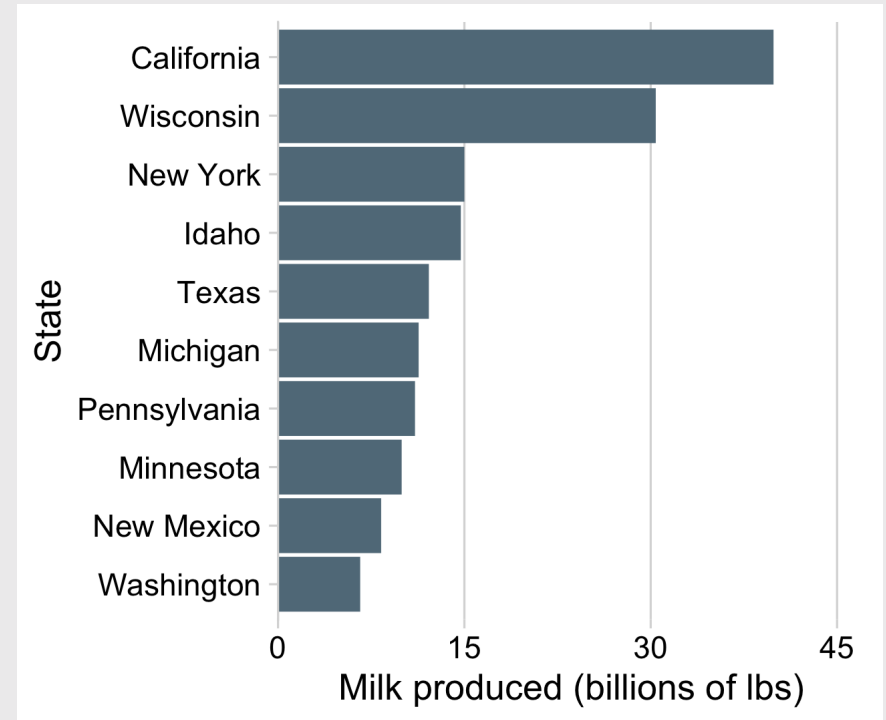
```
milk_bars <- milk_production %>%
 filter(year == 2017) %>%
 arrange(desc(milk_produced)) %>%
 slice(1:10) %>%
 mutate(
 milk_produced = milk_produced / 10^9,
 state = fct_reorder(state, milk_produced)) %>%
 ggplot() +
 geom_col(aes(x = milk_produced, y = state)) +
 theme_minimal_vgrid(font_size = 18) +
 labs(x = 'Milk produced (billions of lbs)',
 y = 'State')

milk_bars
```



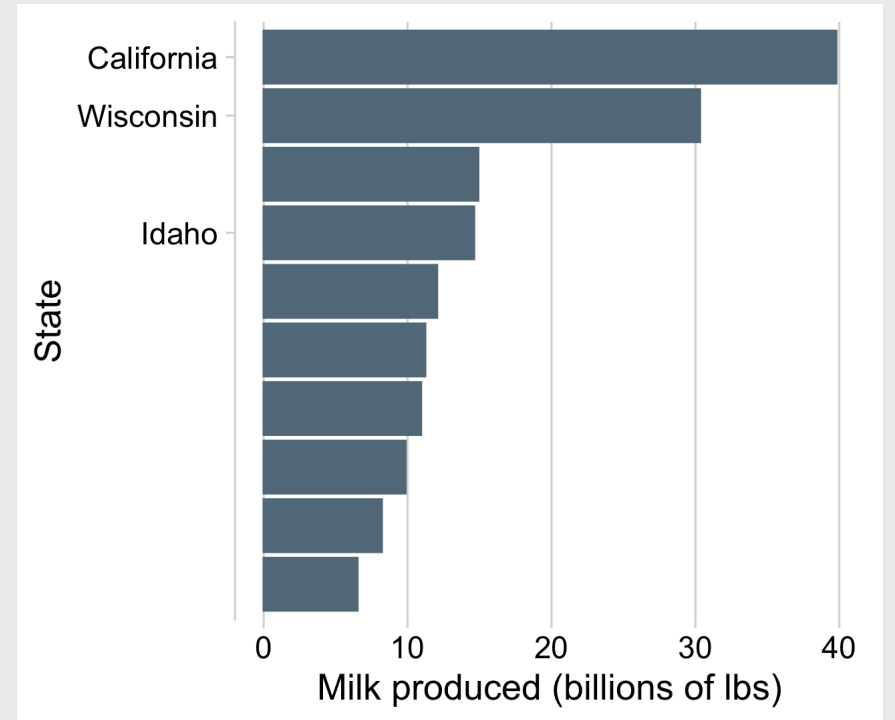
# Adjusting **continuous** scales

```
milkBars +
 scale_x_continuous(
 breaks = c(0, 15, 30, 45),
 limits = c(0, 45),
 expand = expand_scale(mult = c(0, 0.05)))
```



# Adjusting **discrete** scales

```
milkBars +
 scale_y_discrete(
 breaks = c('California', 'Wisconsin', 'Idaho'))
```

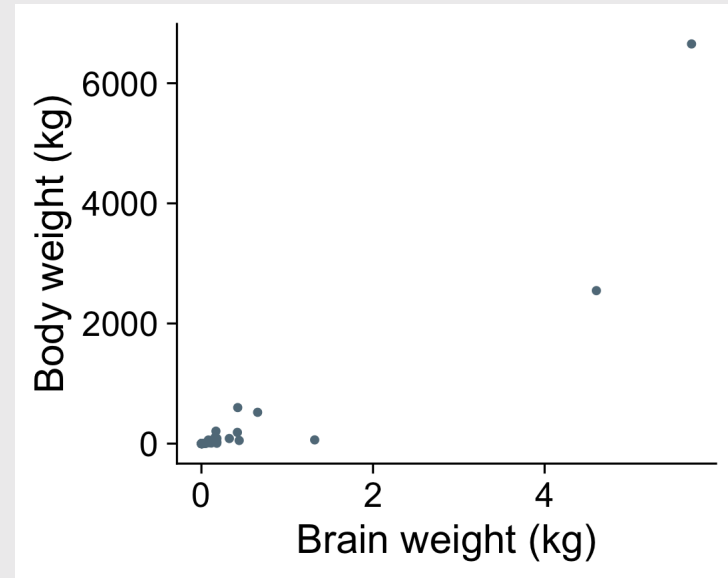


# Adjusting **log** scales

## Regular scaling

```
plot <- ggplot(msleep) +
 geom_point(aes(x = brainwt, y = bodywt)) +
 theme_half_open(font_size = 20) +
 labs(x = 'Brain weight (kg)',
 y = 'Body weight (kg)')
```

plot



# Adjusting **log** scales

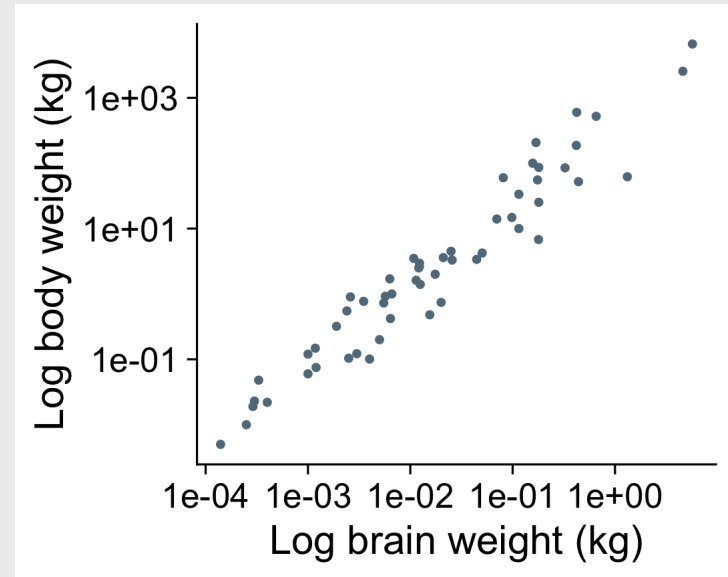
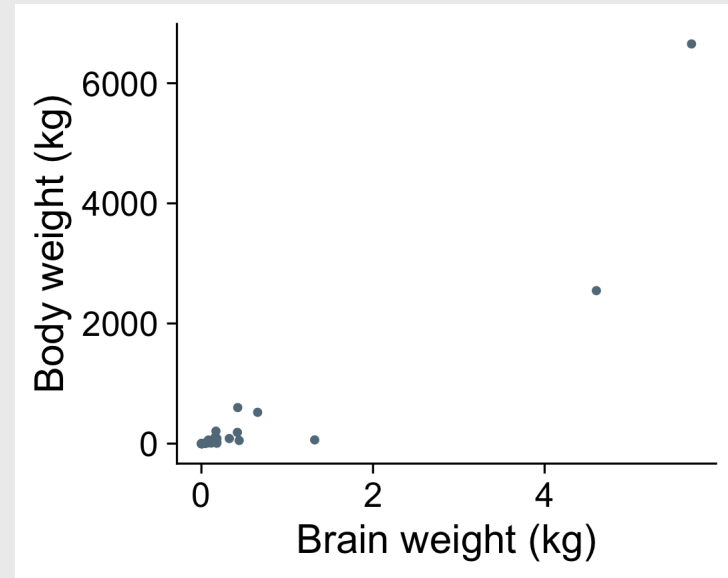
## Log scaling

```
plot +
 scale_x_log10() +
 scale_y_log10() +
 labs(x = 'Log brain weight (kg)',
 y = 'Log body weight (kg)')
```

## Log-log relationship:

$$y = x^n$$

$$\log(y) = n \log(x)$$

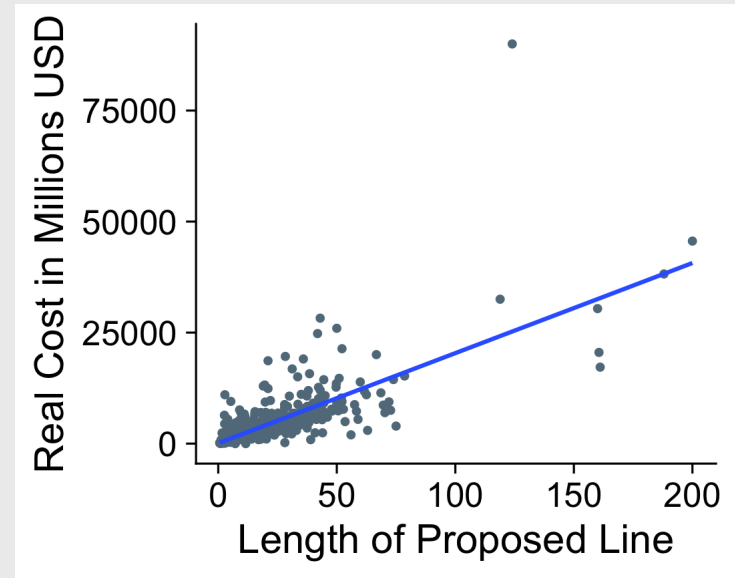


# Example from Mini Project 2

## Regular scaling

```
plot <- transit_cost %>%
 filter(!is.na(length)) %>%
 filter(length < 2500) %>%
 mutate(cost = as.numeric(real_cost)) %>%
 ggplot(aes(x = length, y = cost)) +
 geom_point() +
 geom_smooth(method = 'lm', se = FALSE) +
 theme_half_open(font_size = 20) +
 labs(x = "Length of Proposed Line",
 y = "Real Cost in Millions USD")
```

plot





# Example from Mini Project 2

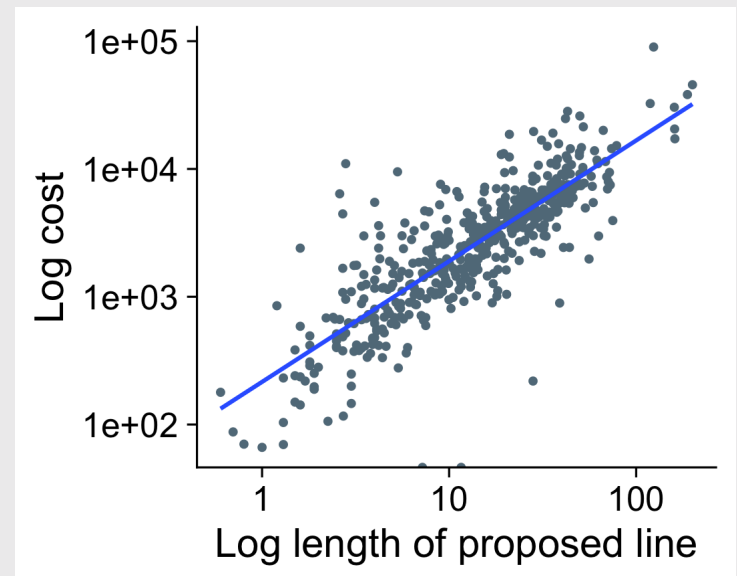
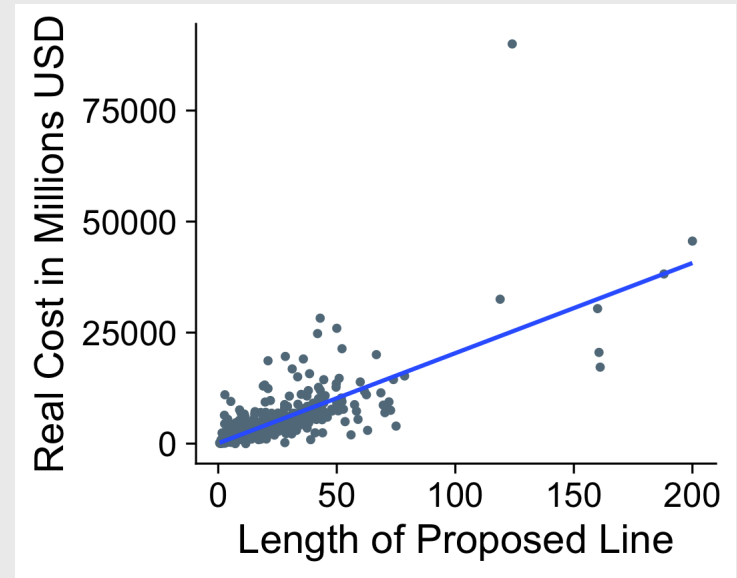
## Log scaling

```
plot +
 scale_x_log10() +
 scale_y_log10() +
 labs(x = 'Log length of proposed line',
 y = 'Log cost')
```

## Log-log relationship:

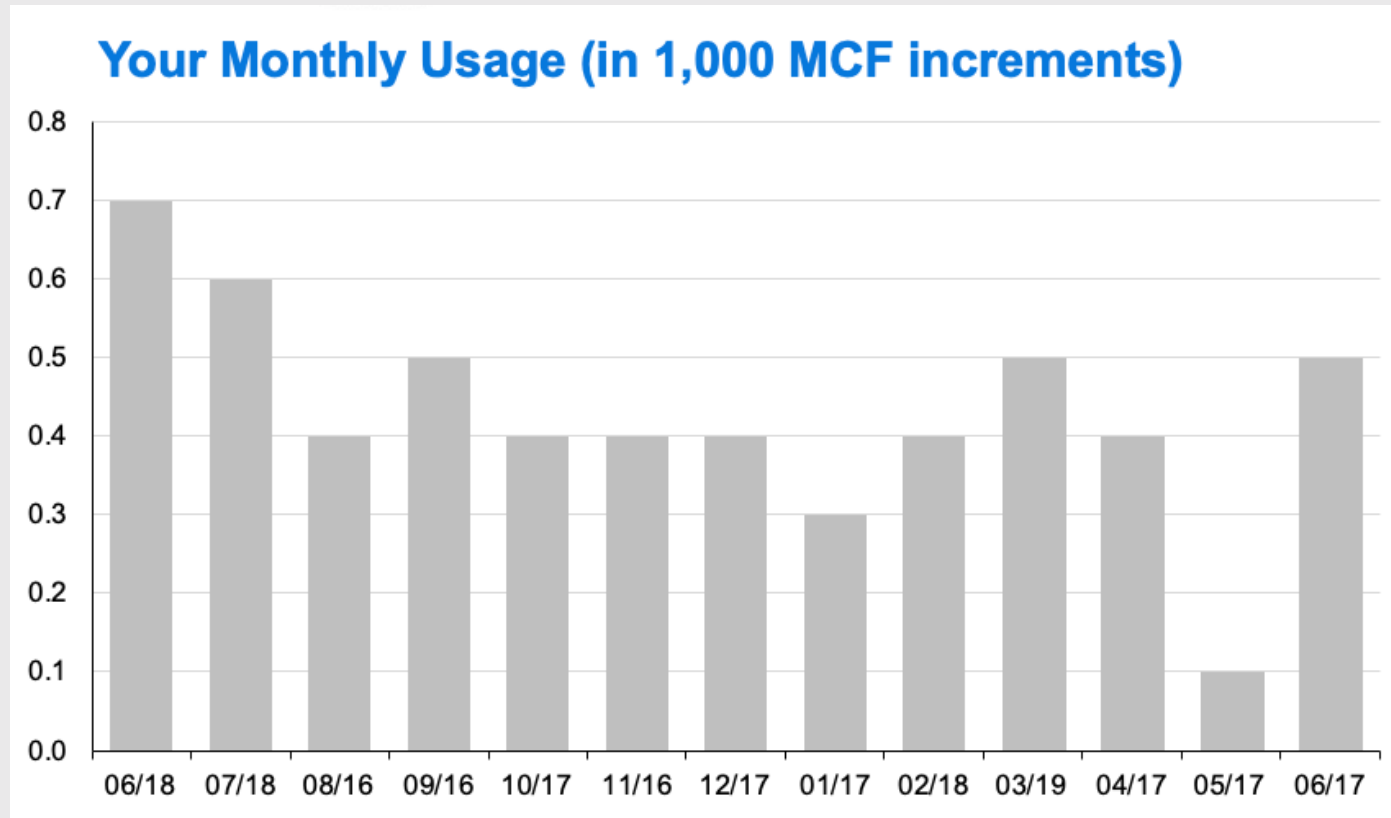
$$y = x^n$$

$$\log(y) = n \log(x)$$

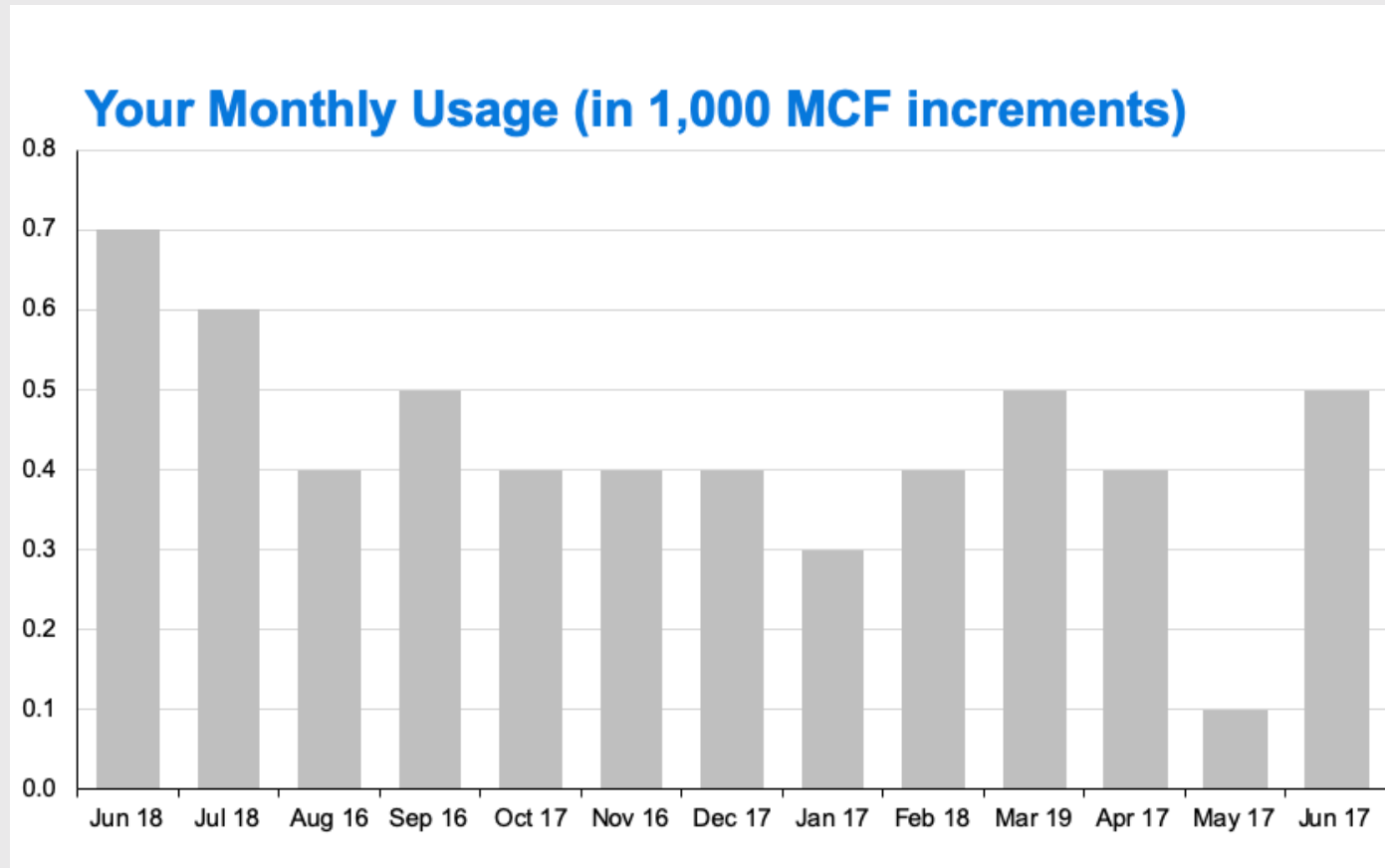


Date scales can be confusing

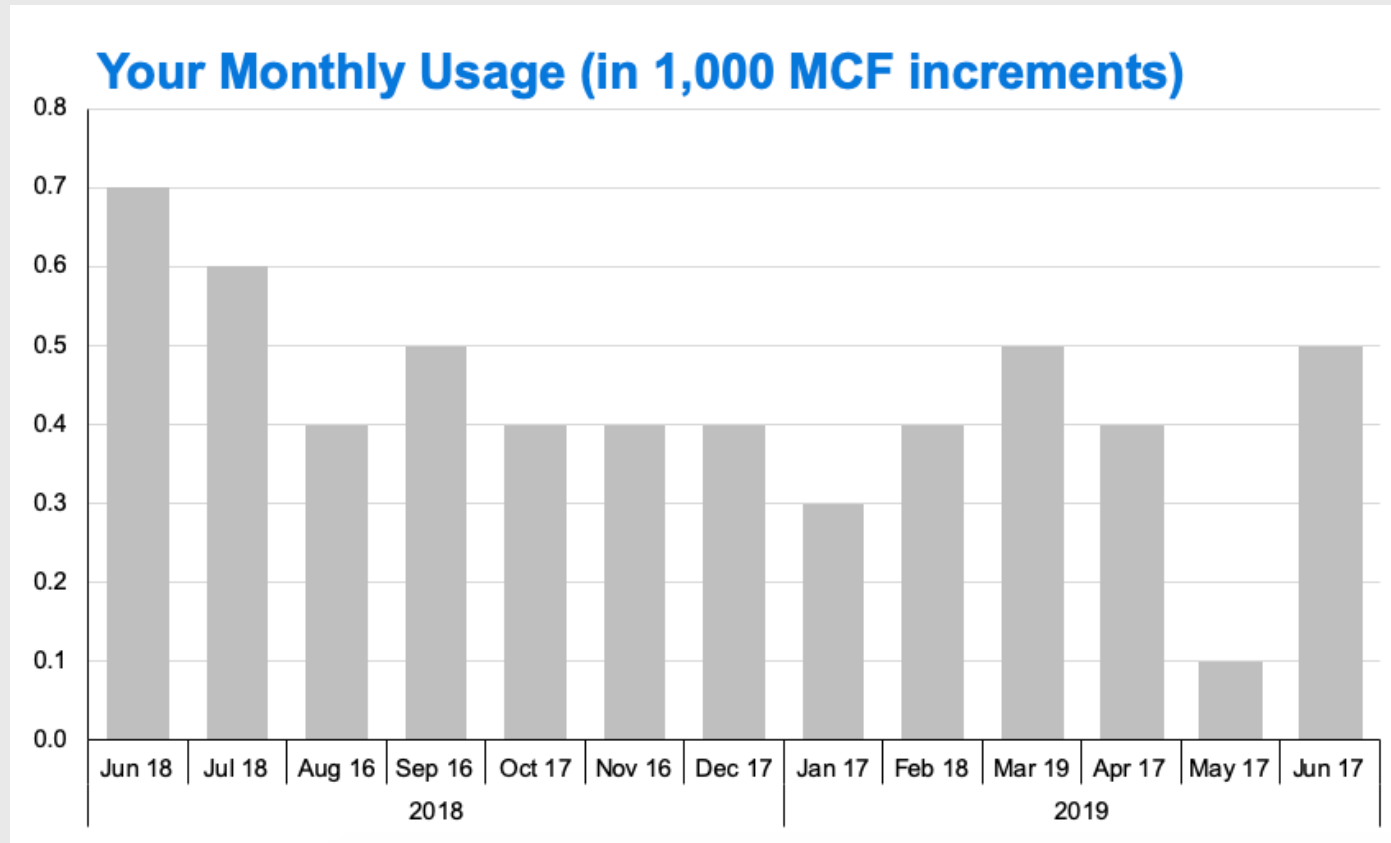
# What's wrong with this chart?



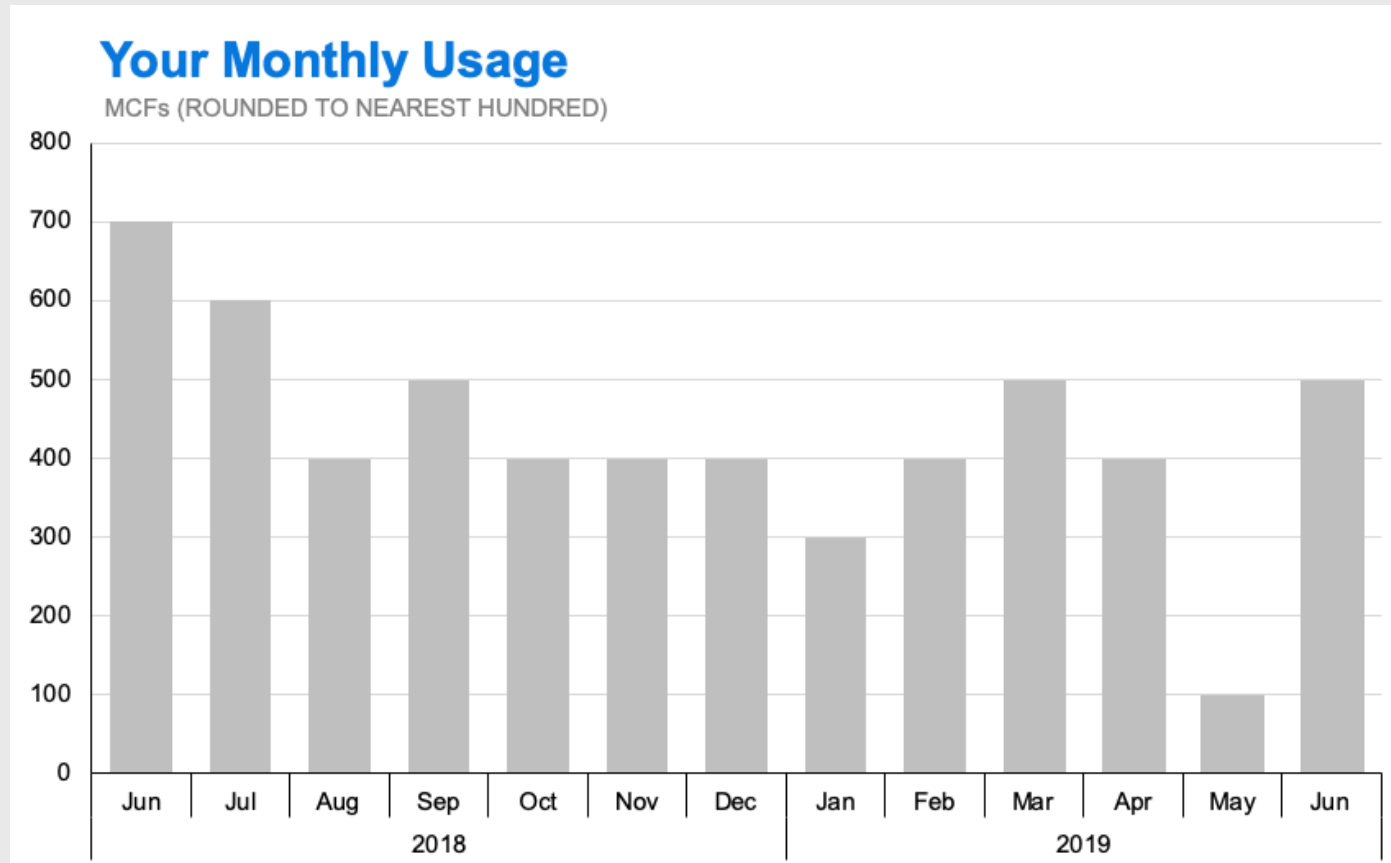
# What's wrong with this chart?



# What's wrong with this chart?



# What's wrong with this chart?



# Adjusting **date** scales

Summarise the data

```
library(lubridate)

plot <- wildlife_impacts %>%
 filter(incident_year == 2016) %>%
 count(operator, incident_date) %>%
 mutate(incident_date = ymd(incident_date)) %>%
 ggplot() +
 geom_col(
 aes(x = incident_date, y = n,
 color = operator)) +
 facet_wrap(~operator, ncol = 1) +
 theme_minimal_grid(font_size = 16) +
 panel_border() +
 theme(legend.position = 'none') +
 labs(x = 'Incident date (2016)',
 y = 'Number of incidents')

plot
```

# Adjusting **date** scales

```
plot +
 scale_x_date(
 date_breaks = '1 month',
 date_labels = '%b')
```



# Adjusting **date** scales

```
scale_x_date(
 date_breaks = '1 month',
 date_labels = '%b')
```

```
date_breaks = '1 month'
```

- '1 day'
- '10 days'
- '1 month'
- '3 months'
- '1 year'
- '3 years'

```
date_labels = '%b'
```

Example date: March 04, 2020

- %Y = 2020
- %y = 20
- %B = March
- %b = Mar
- %D = 03/04/2020
- %d = 03

Use **scales** library to modify scale **text**

# scales converts numbers to formatted characters

```
scales::comma(200000)
```

```
#> [1] "200,000"
```

```
scales::dollar(200000)
```

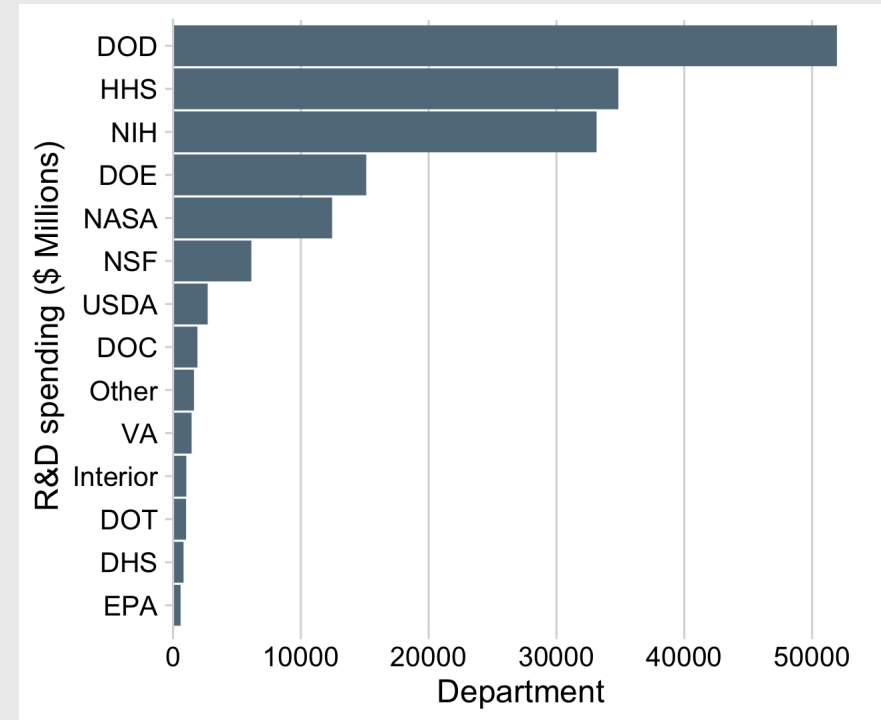
```
#> [1] "$200,000"
```

```
scales::percent(0.5)
```

```
#> [1] "50%"
```

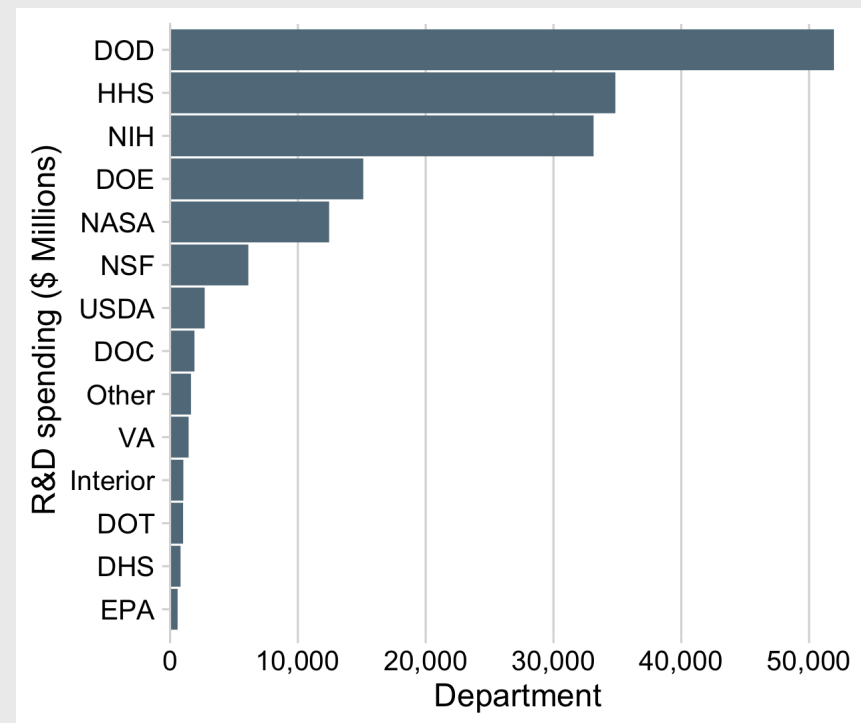
# Use **scales** library to modify scale text

```
federal_spending %>%
 filter(year == 2017) %>%
 mutate(
 department = fct_reorder(
 department, rd_budget)) %>%
 ggplot() +
 geom_col(aes(x = rd_budget, y = department))
 scale_x_continuous(
 expand = expand_scale(mult = c(0, 0.05)))
 theme_minimal_vgrid(font_size = 16) +
 labs(x = 'Department',
 y = 'R&D spending ($ Millions)')
```



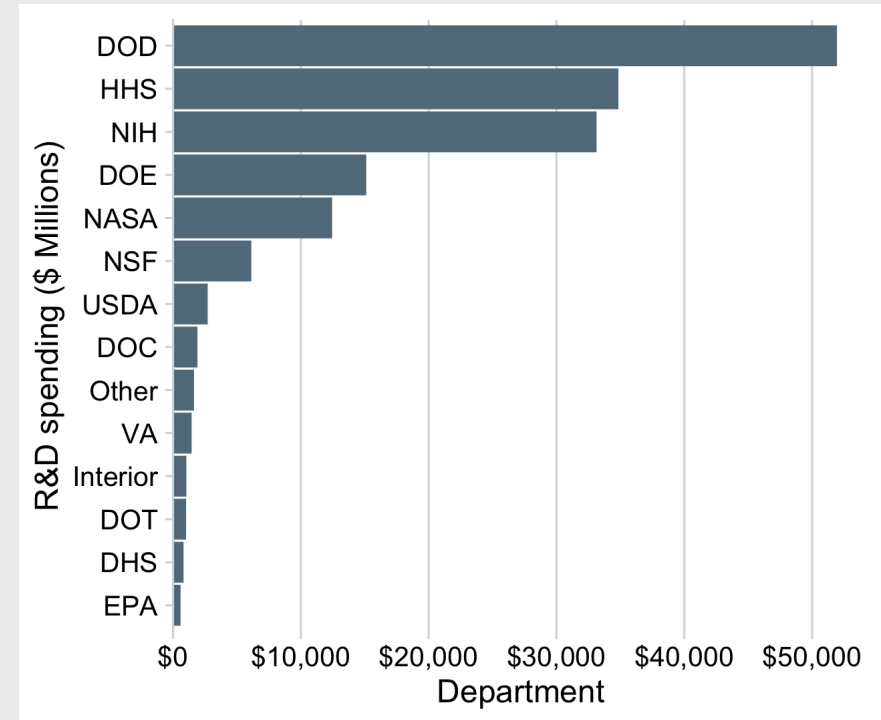
# Use **scales** library to modify scale text

```
federal_spending %>%
 filter(year == 2017) %>%
 mutate(
 department = fct_reorder(
 department, rd_budget)) %>%
 ggplot() +
 geom_col(aes(x = rd_budget, y = department))
 scale_x_continuous(
 labels = scales::comma,
 expand = expand_scale(mult = c(0, 0.05)))
 theme_minimal_vgrid(font_size = 16) +
 labs(x = 'Department',
 y = 'R&D spending ($ Millions)')
```



# Use **scales** library to modify scale text

```
federal_spending %>%
 filter(year == 2017) %>%
 mutate(
 department = fct_reorder(
 department, rd_budget)) %>%
 ggplot() +
 geom_col(aes(x = rd_budget, y = department))
 scale_x_continuous(
 labels = scales::dollar,
 expand = expand_scale(mult = c(0, 0.05)))
 theme_minimal_vgrid(font_size = 16) +
 labs(x = 'Department',
 y = 'R&D spending ($ Millions)')
```

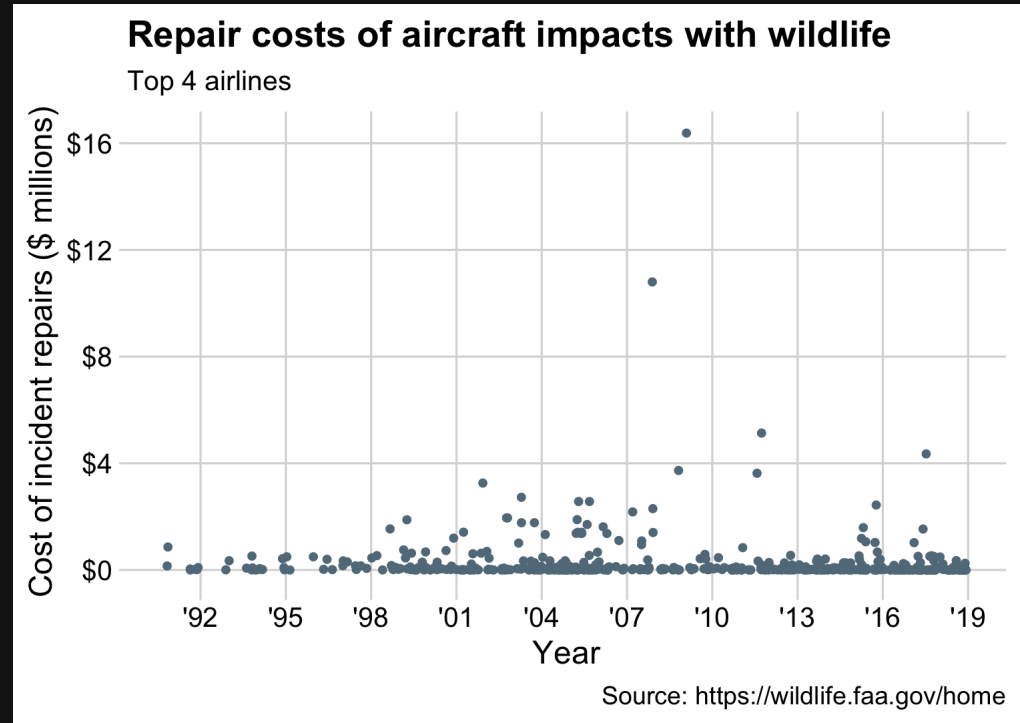


Check out this guide to scales:  
<https://ggplot2tor.com/scales>

# Your turn

15:00

Adjust the scales in the code chunk provided to match the chart on the slides.





# Week 10: *Polishing Charts*

1. Scales

2. Annotations

BREAK

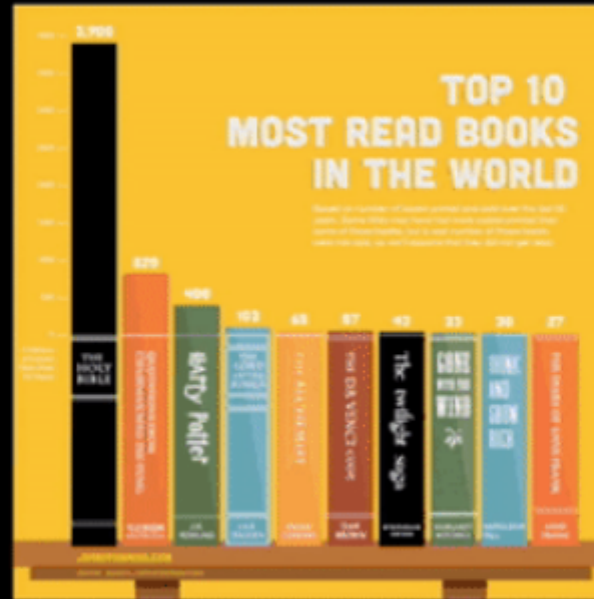
3. Colors

4. Fonts

5. Rmd tricks

Text is usually the single most important component on your chart

# STEP I: Encoding



10 SECONDS

● = eye fixation

Michelle Borkin, et al. (2015) [Beyond Memorability: Visualization Recognition and Recall](#)

# Are Americans Falling Out of Love With the Cinema?

Estimated number of tickets sold at the North American box office since 1995



Source: The Numbers



statista

Titles matter

Source:

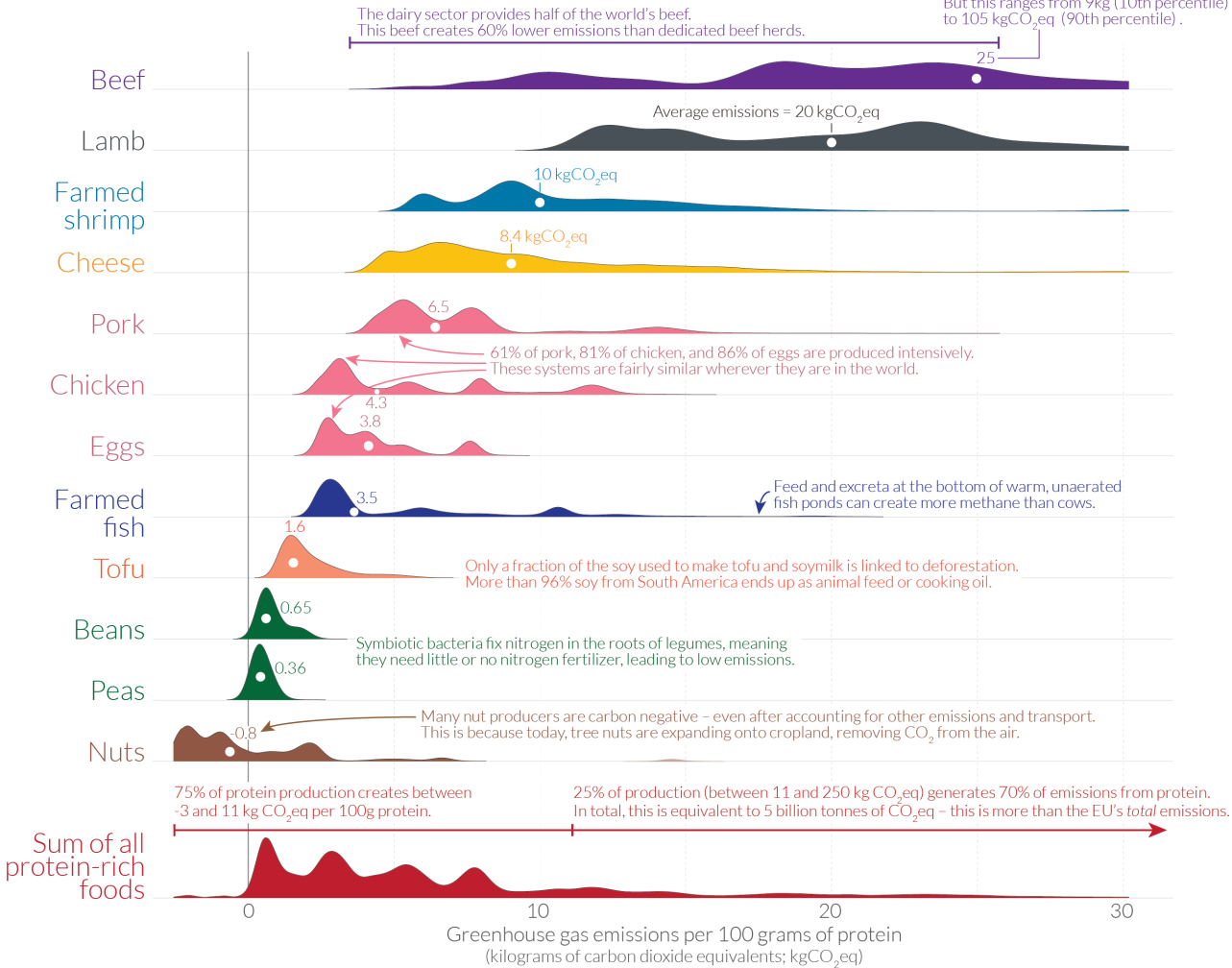
<https://www.reddit.com/r/dataisugly/co>

# How does the carbon footprint of protein-rich foods compare?



Greenhouse gas emissions from protein-rich foods are shown per 100 grams of protein across a global sample of 38,700 commercially viable farms in 119 countries. The height of the curve represents the amount of production globally with that specific footprint. The white dot marks the median greenhouse gas emissions for each food product.

Producing 100 grams of protein from beef emits 25 kilograms of CO<sub>2</sub>eq, on average. But this ranges from 9kg (10th percentile) to 105 kgCO<sub>2</sub>eq (90th percentile).

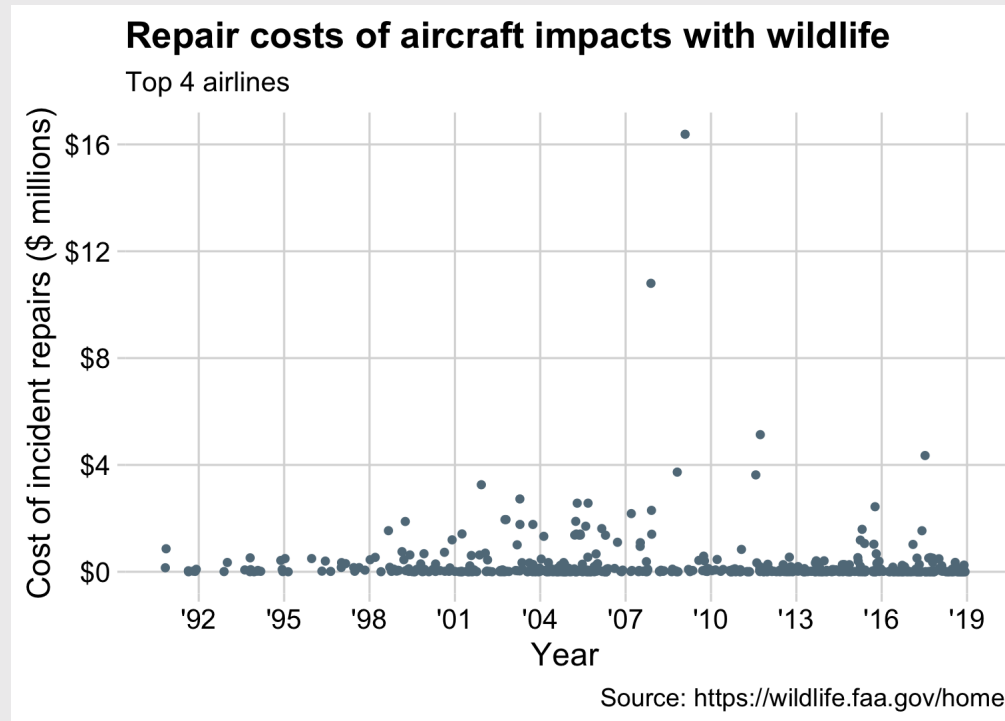


Note: Data refers to the greenhouse gas emissions of food products across a global sample of 38,700 commercially viable farms in 119 countries. Emissions are measured across the full supply-chain, from land use change through to the retailer and includes on-farm, processing, transport, packaging and retail emissions. Data source: Joseph Poore and Thomas Nemecek (2018). Reducing food's environmental impacts through producers and consumers. *Science*. OurWorldinData.org – Research and data to make progress against the world's largest problems. Licensed under CC-BY by the authors Joseph Poore & Hannah Ritchie.

## Good annotations should tell a story

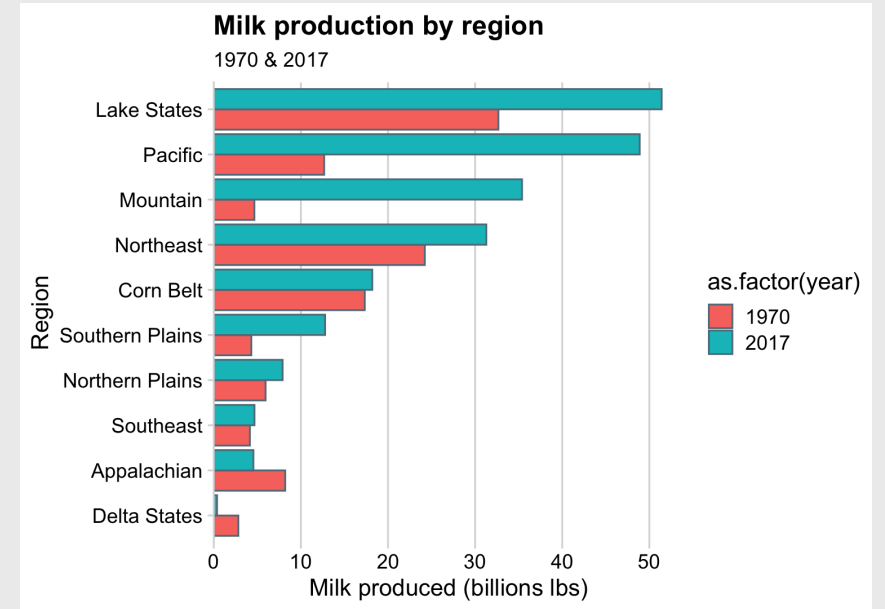
Source:  
<https://ourworldindata.org/less-meat-or-sustainable-meat>

```
labs(
 x = 'Year',
 y = 'Cost of incident repairs ($ millions)',
 title = 'Repair costs of aircraft impacts with wildlife',
 subtitle = 'Top 4 airlines',
 caption = 'Source: https://wildlife.faa.gov/home')
```



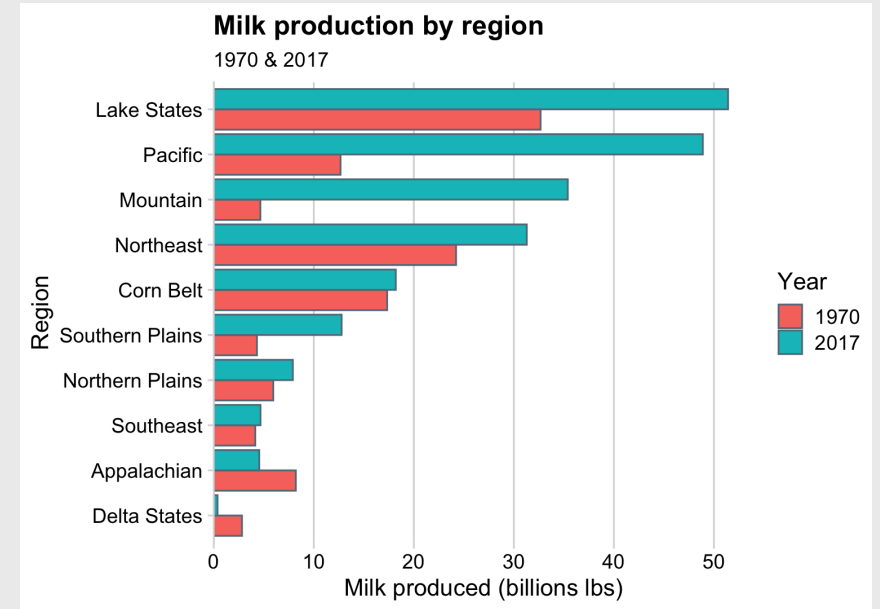
## Use mapped variables in `aes()` in `labs()`

```
milk_production %>%
 filter(year %in% c(1970, 2017)) %>%
 group_by(year, region) %>%
 summarise(milk_produced = sum(milk_produced) / 10^9) %>%
 ungroup() %>%
 mutate(
 region = fct_reorder2(region, year, desc(milk_produced))
)
ggplot() +
 geom_col(
 aes(x = milk_produced,
 y = region,
 fill = as.factor(year)),
 position = "dodge") +
 scale_x_continuous(expand = expansion(mult = c(0, 0.05))) +
 theme_minimal_vgrid() +
 labs(
 x = 'Milk produced (billions lbs)',
 y = 'Region',
 title = 'Milk production by region',
 subtitle = '1970 & 2017')
```



## Use mapped variables in `aes()` in `labs()`

```
milk_production %>%
 filter(year %in% c(1970, 2017)) %>%
 group_by(year, region) %>%
 summarise(milk_produced = sum(milk_produced) / 10^9) %>%
 ungroup() %>%
 mutate(
 region = fct_reorder2(region, year, desc(milk_produced))
)
ggplot() +
 geom_col(
 aes(x = milk_produced,
 y = region,
 fill = as.factor(year)),
 position = "dodge") +
 scale_x_continuous(expand = expansion(mult = c(0, 0.05))) +
 theme_minimal_vgrid() +
 labs(
 x = 'Milk produced (billions lbs)',
 y = 'Region',
 title = 'Milk production by region',
 subtitle = '1970 & 2017',
 fill = "Year")
```



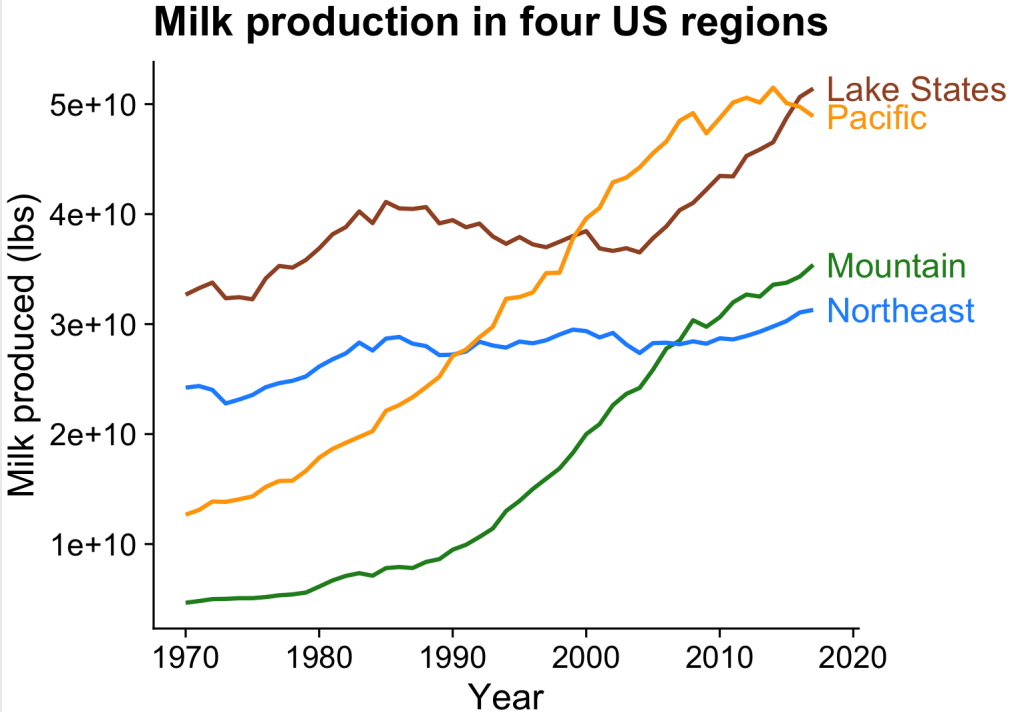
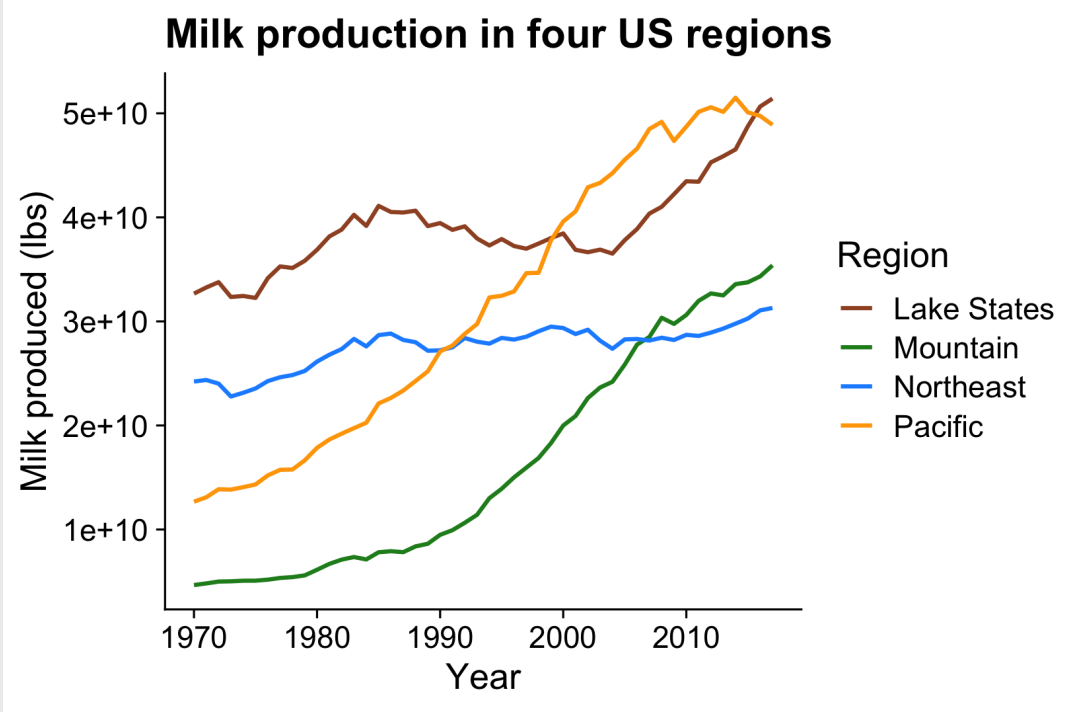


Legends suck

# Legends suck

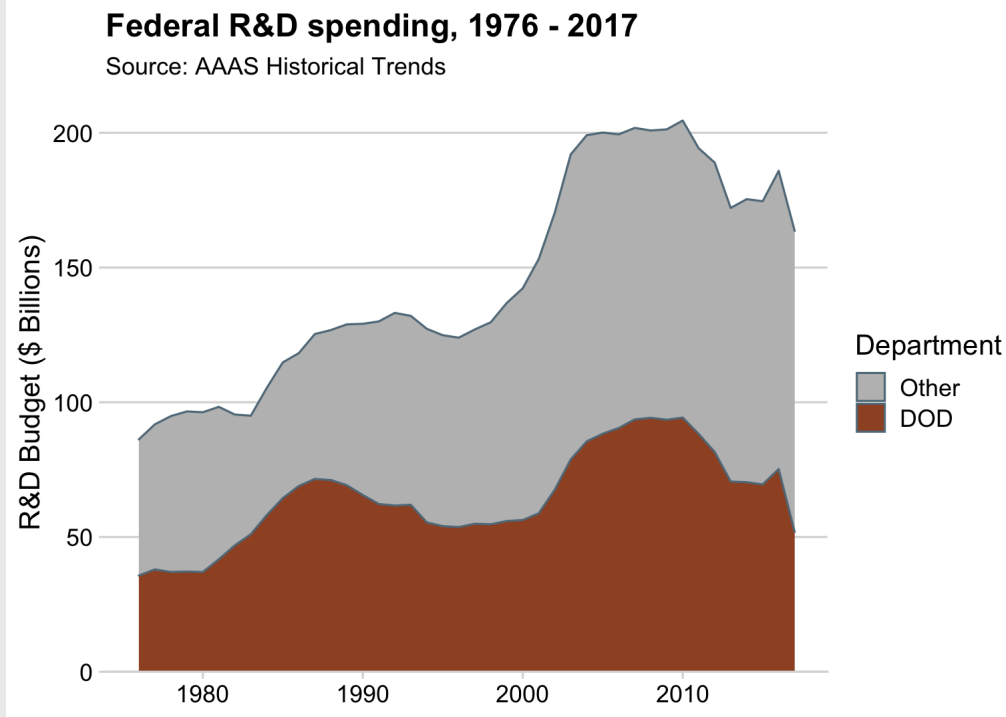
Legends require look-up task

Direct labeling is much better

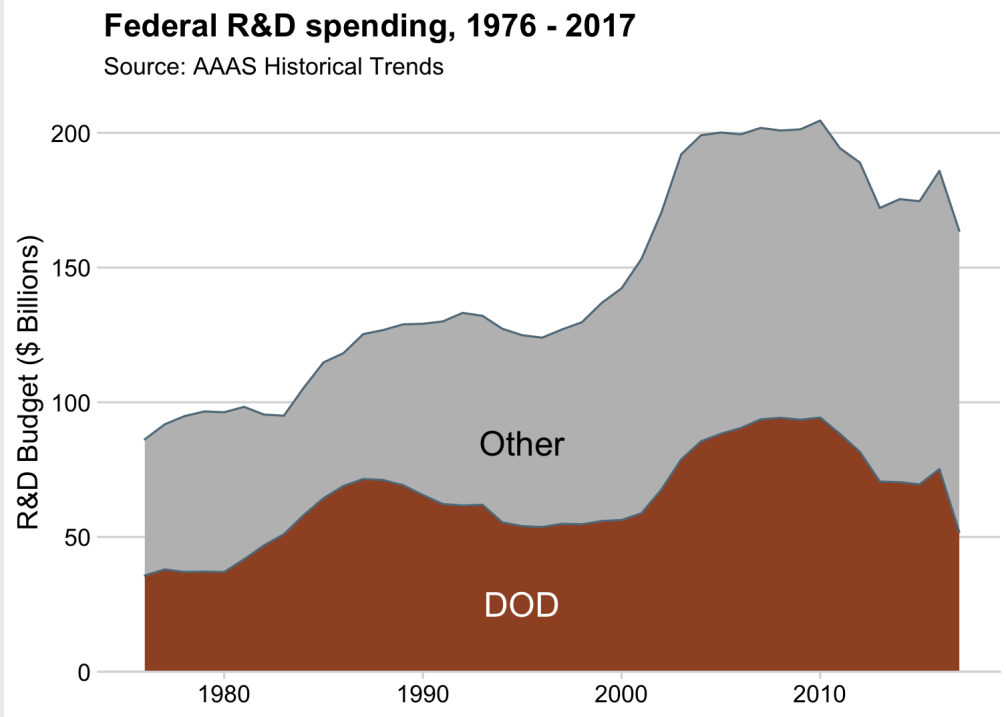


# Legends suck

Legends require look-up task



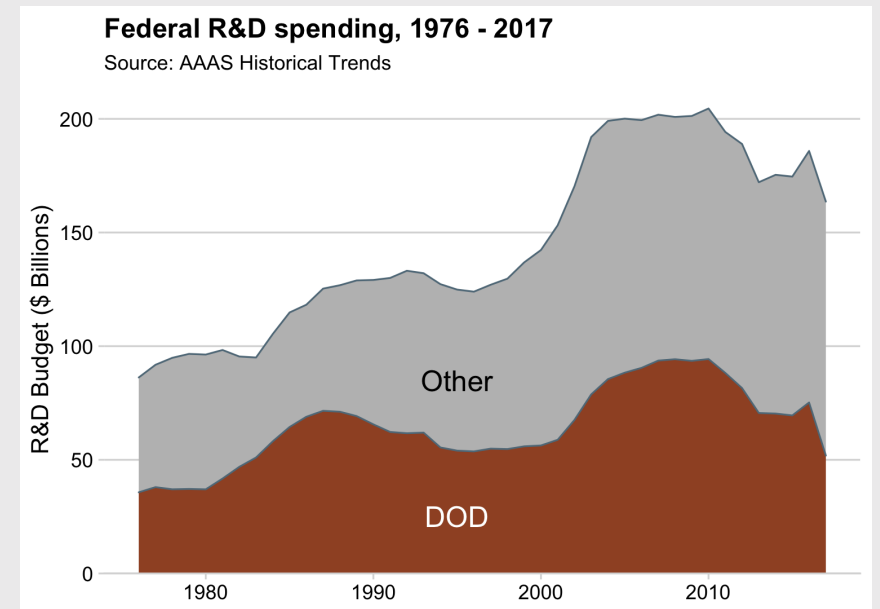
Direct labeling is much better



# Use `annotate()` to add text to chart

```
dod_spending <- ggplot(federal_spending_summary) +
 geom_area(aes(x = year, y = rd_budget,
 fill = department)) +
 annotate(geom = 'text', x = 1995, y = 85,
 label = 'Other', size = 6, color = 'black')
 annotate(geom = 'text', x = 1995, y = 25,
 label = 'DOD', size = 6, color = 'white') +
 scale_y_continuous(
 expand = expand_scale(mult = c(0, 0.05))) +
 scale_fill_manual(values = c('grey', 'sienna')) +
 theme_minimal_hgrid() +
 theme(legend.position = 'none') +
 labs(x = NULL,
 y = 'R&D Budget ($ Billions)',
 title = 'Federal R&D spending, 1976 - 2017',
 subtitle = 'Source: AAAS Historical Trends')
```

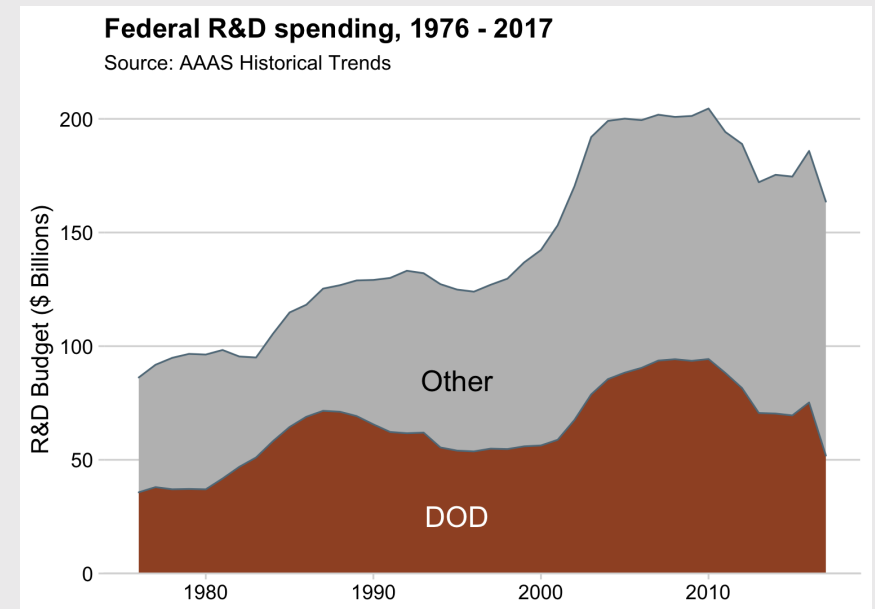
dod\_spending



# Use `geom_text()` to add text to chart

```
dod_spending <- ggplot(federal_spending_summary) +
 geom_area(aes(x = year, y = rd_budget,
 fill = department)) +
 geom_text(
 data = data.frame(x = 1995, y = 85, label = 'Other'),
 aes(x = x, y = y, label = label),
 size = 6, color = 'black') +
 geom_text(
 data = data.frame(x = 1995, y = 25, label = 'DOD'),
 aes(x = x, y = y, label = label),
 size = 6, color = 'white') +
 scale_y_continuous(
 expand = expand_scale(mult = c(0, 0.05))) +
 scale_fill_manual(values = c('grey', 'sienna')) +
 theme_minimal_hgrid() +
 theme(legend.position = 'none') +
 labs(x = NULL,
 y = 'R&D Budget ($ Billions)',
 title = 'Federal R&D spending, 1976 - 2017',
 subtitle = 'Source: AAAS Historical Trends')
```

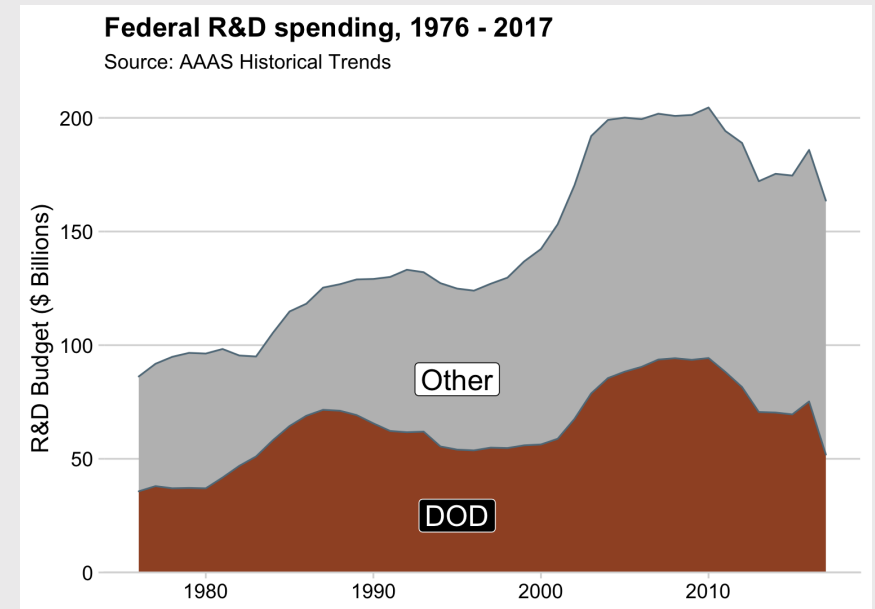
```
dod_spending
```



# Use `geom_label()` to add text to chart **with a background**

```
dod_spending <- ggplot(federal_spending_summary) +
 geom_area(aes(x = year, y = rd_budget,
 fill = department)) +
 geom_label(
 data = data.frame(x = 1995, y = 85, label = 'Other'),
 aes(x = x, y = y, label = label),
 size = 6) +
 geom_label(
 data = data.frame(x = 1995, y = 25, label = 'DOD'),
 aes(x = x, y = y, label = label),
 size = 6, fill = "black", color = "white") +
 scale_y_continuous(
 expand = expand_scale(mult = c(0, 0.05))) +
 scale_fill_manual(values = c('grey', 'sienna')) +
 theme_minimal_hgrid() +
 theme(legend.position = 'none') +
 labs(x = NULL,
 y = 'R&D Budget ($ Billions)',
 title = 'Federal R&D spending, 1976 - 2017',
 subtitle = 'Source: AAAS Historical Trends')
```

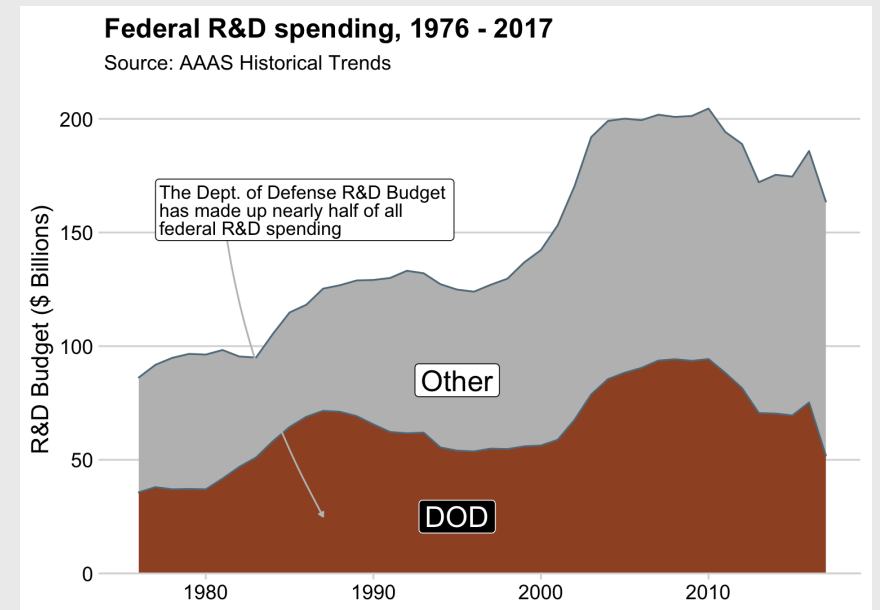
```
dod_spending
```



# Use `geom_curve()` + `geom_label()` to direct attention

```
label <- "The Dept. of Defense R&D Budget
has made up nearly half of all
federal R&D spending"

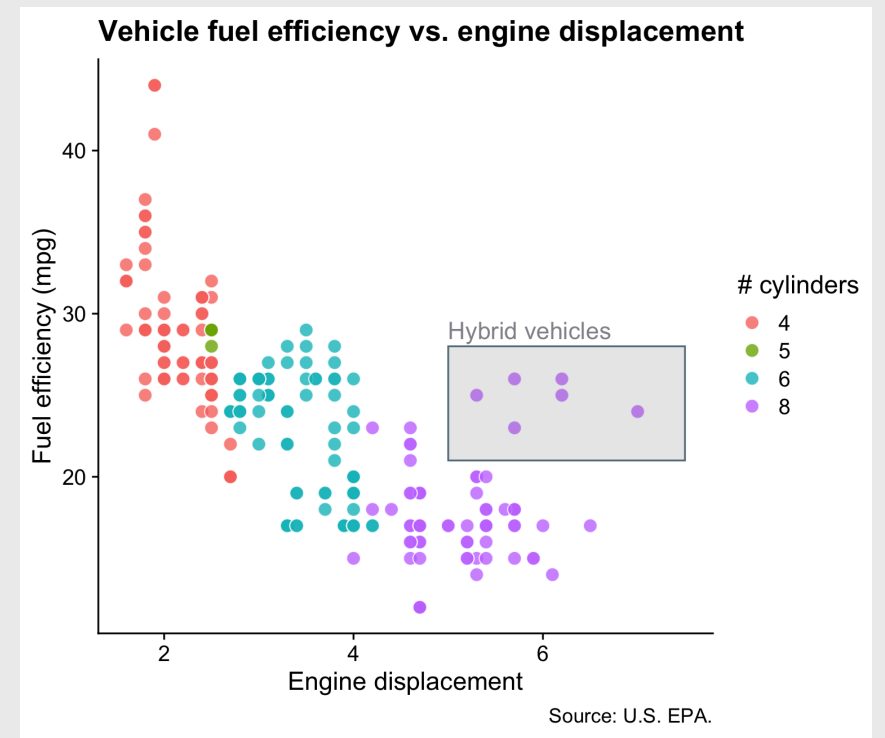
dod_spending +
 geom_curve(
 data = data.frame(
 x = 1981, xend = 1987, y = 160, yend = 25),
 mapping = aes(x = x, xend = xend, y = y, yend = yend,
 color = 'grey75', size = 0.5, curvature = 0.1,
 arrow = arrow(length = unit(0.01, "npc"),
 type = "closed")) +
 geom_label(
 data = data.frame(x = 1977, y = 160, label = label),
 mapping = aes(x = x, y = y, label = label),
 hjust = 0, lineheight = 0.8)
```



# Use `annotate()` to direct attention

Use `geom = "rect"` for box, `geom = "text"` for label

```
ggplot(mpg, aes(x = displ, y = hwy)) +
 geom_point(
 aes(fill = as.factor(cyl)),
 color = 'white', alpha = 0.8,
 size = 3.5, shape = 21) +
 annotate(geom = "rect",
 xmin = 5, xmax = 7.5,
 ymin = 21, ymax = 28,
 fill = "grey55", alpha = 0.2) +
 annotate(geom = "text",
 x = 5, y = 29, label = "Hybrid vehicles",
 hjust = 0, size = 5) +
 theme_half_open(font_size = 15) +
 labs(x = "Engine displacement",
 y = "Fuel efficiency (mpg)",
 fill = '# cylinders',
 title = "Vehicle fuel efficiency vs. engine displ",
 caption = "Source: U.S. EPA.")
```





# Find where to put annotations with `ggannotate`

Install:

```
remotes::install_github("mattcowgill/ggannotate")
```

Use:

```
library(ggannotate)

plot <- ggplot(mpg) +
 geom_point(aes(x = displ, y = hwy, color = as.factor(cy
 theme_half_open()

ggannotate(plot)
```

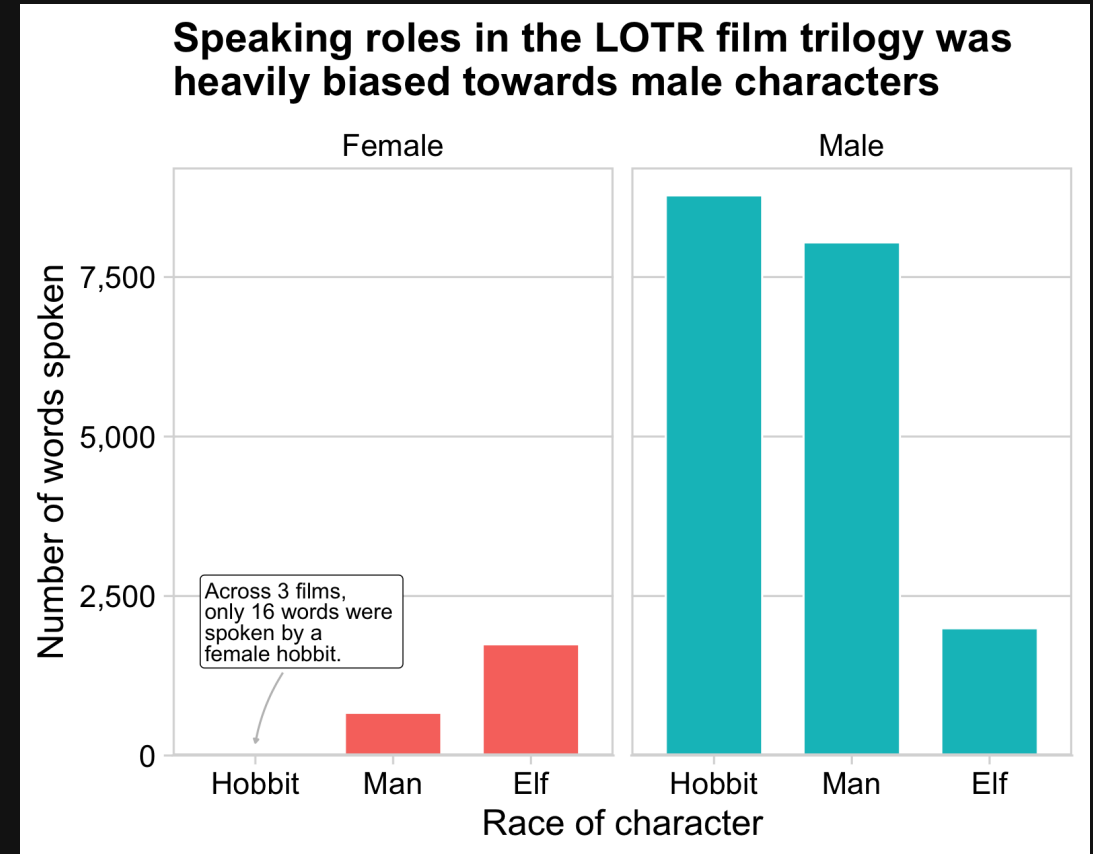
# Your turn

20:00

Use the `lotr_summary` data frame to create the following chart.

Hints:

- For the `geom_label()`, use these points:
  - `x = 0.6`
  - `y = 2100`
- For the `geom_curve()`, use these points:
  - `x = 1.2`
  - `xend = 1`
  - `y = 1300`
  - `yend = 200`



# Intermission

05:00

# Week 10: *Polishing Charts*

1. Scales

2. Annotations

BREAK

3. Colors

4. Fonts

5. Rmd tricks

Color is hard

# How do I know what colors look good together?

Use the color wheel



Image from [this color wheel tool](#)

# How do I know what colors look good together?

Use the color wheel

1. **Complementary:** High contrast

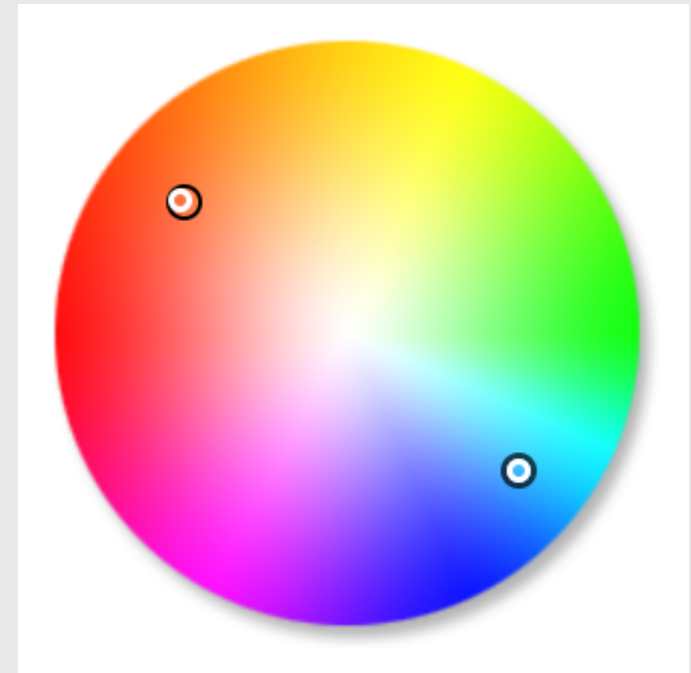


Image from [this color wheel tool](#)

# How do I know what colors look good together?

Use the color wheel

1. Complementary: High contrast
2. **Analogous**: Calm, harmonious

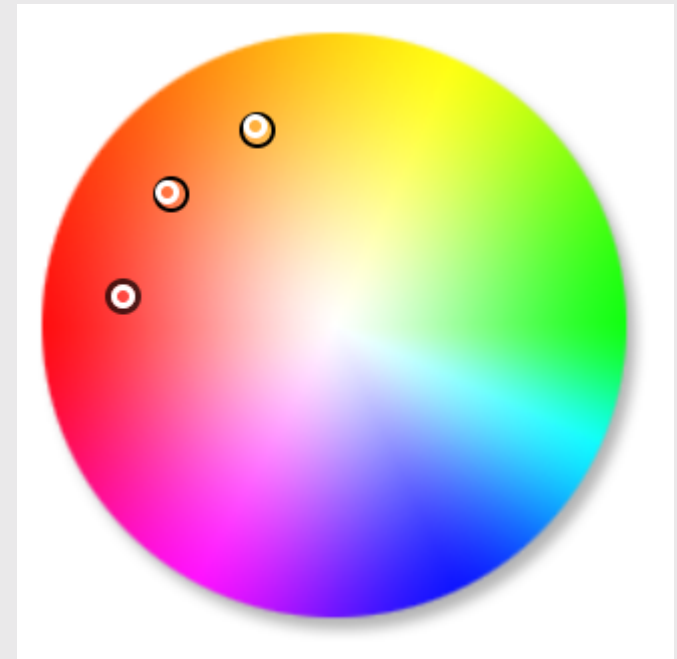


Image from [this color wheel tool](#)



# How do I know what colors look good together?

Use the color wheel

1. Complementary: High contrast
2. Analogous: Calm, harmonious
3. **Triadic**: Vibrant, contrast

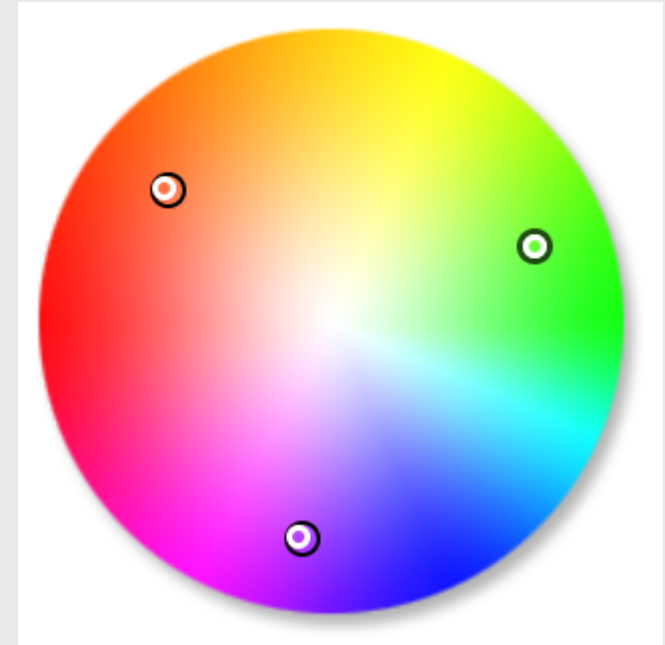


Image from [this color wheel tool](#)

Artists use color theory too!

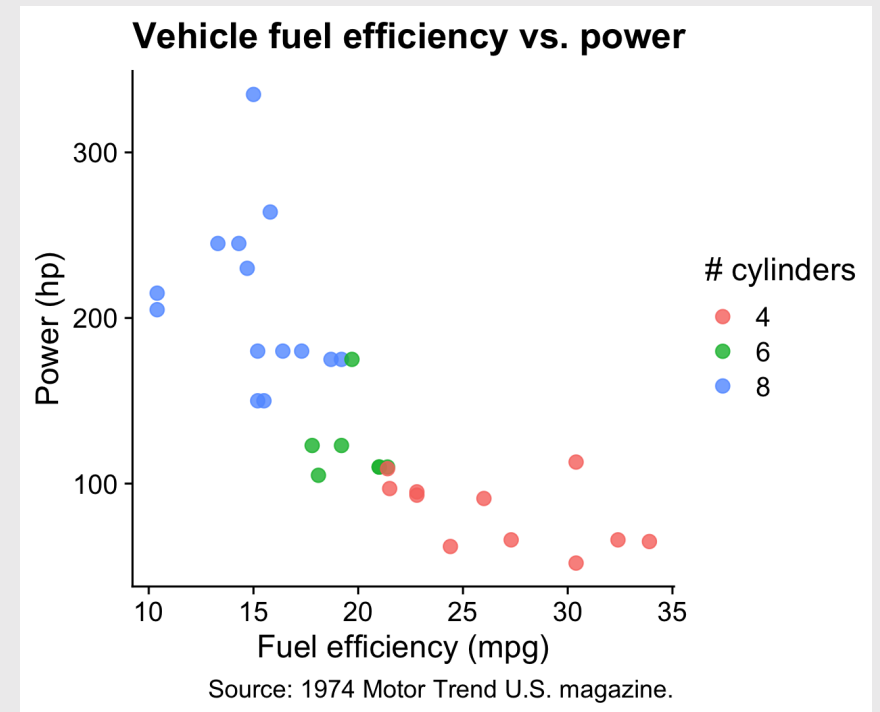
Steal colors with the **eye dropper tool**



# Using your own colors

Map color to variable

```
mpg_plot <- ggplot(mtcars, aes(x = mpg, y = hp)) +
 geom_point(
 aes(color = as.factor(cyl)),
 alpha = 0.8, size = 3) +
 theme_half_open(font_size = 16) +
 labs(x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 color = "# cylinders",
 title = "Vehicle fuel efficiency vs. power",
 caption = "Source: 1974 Motor Trend U.S. magazine")
```



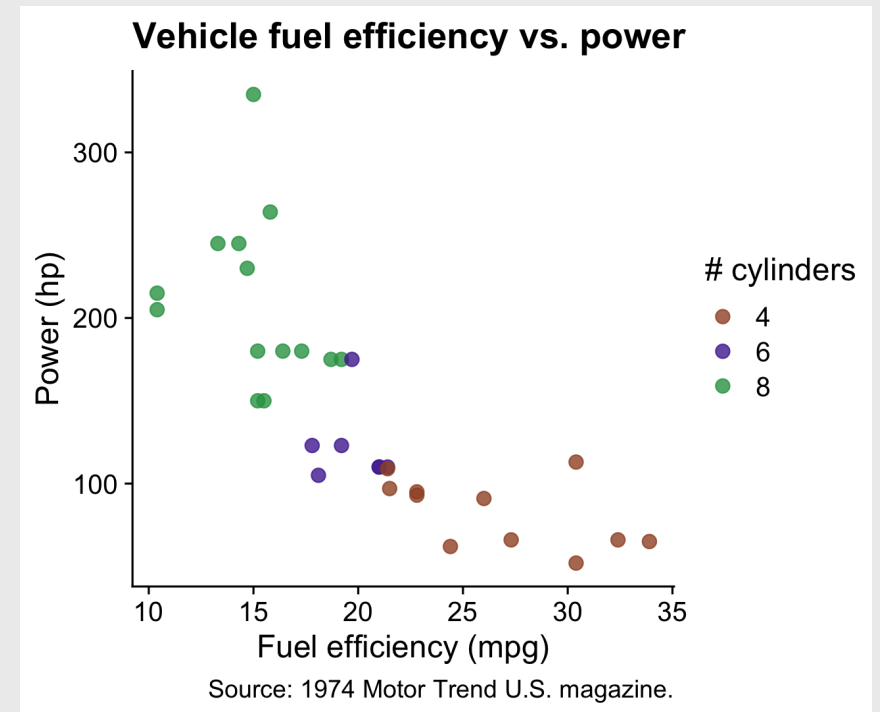
# Using your own colors

## Map color to variable

```
mpg_plot <- ggplot(mtcars, aes(x = mpg, y = hp)) +
 geom_point(
 aes(color = as.factor(cyl)),
 alpha = 0.8, size = 3) +
 theme_half_open(font_size = 16) +
 labs(x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 color = "# cylinders",
 title = "Vehicle fuel efficiency vs. power",
 caption = "Source: 1974 Motor Trend U.S. magazine")
```

## Manually change colors

```
mpg_plot +
 scale_color_manual(values = c(
 '#a0522d', '#522da0', '#2da052'))
```



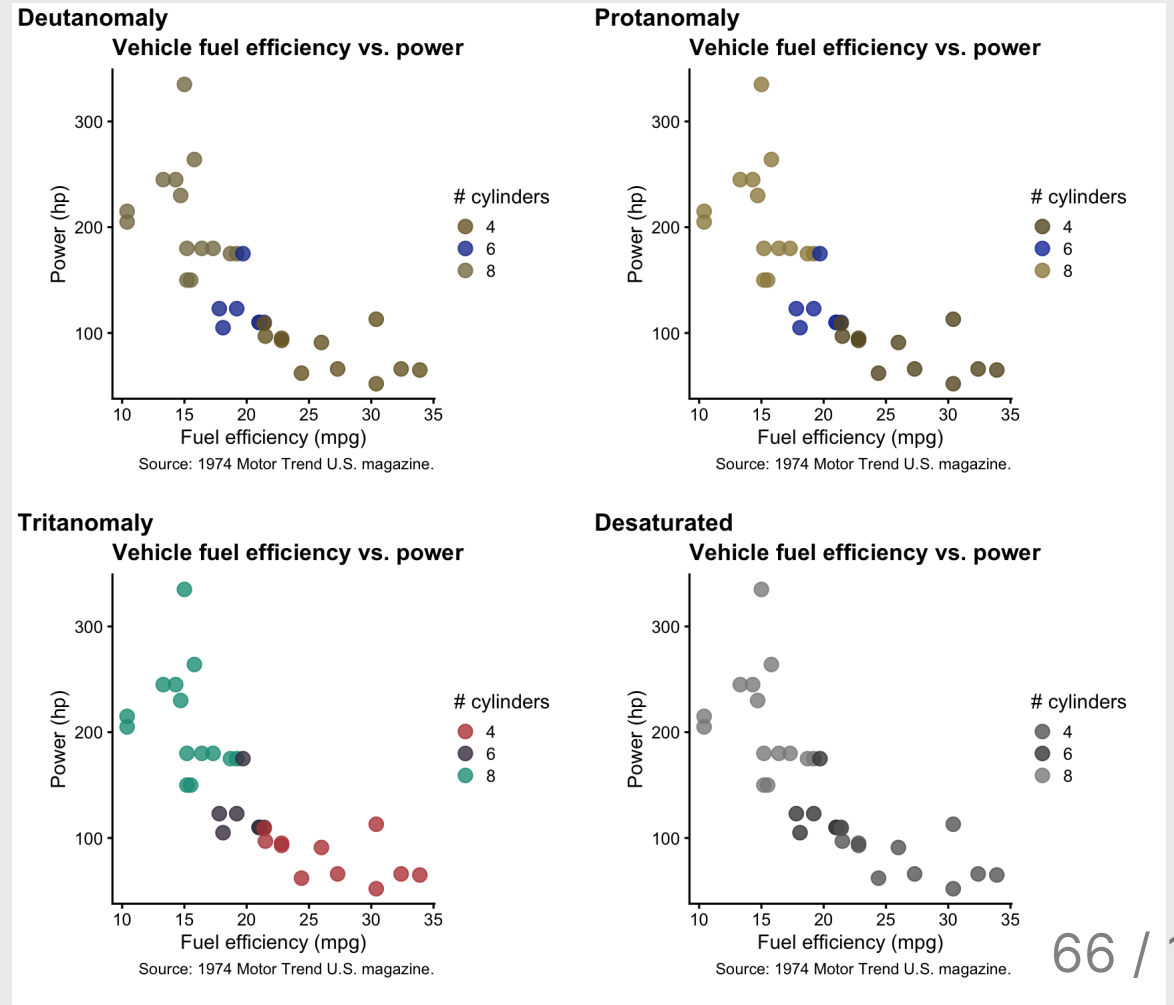
# Consider color blind friendly colors

## Manually change colors

```
mpg_plot_mycolors <- mpg_plot +
 theme_half_open(font_size = 10) +
 scale_color_manual(values = c(
 '#a0522d', '#522da0', '#2da052'))
```

## Simulate color blindness with colorblindr

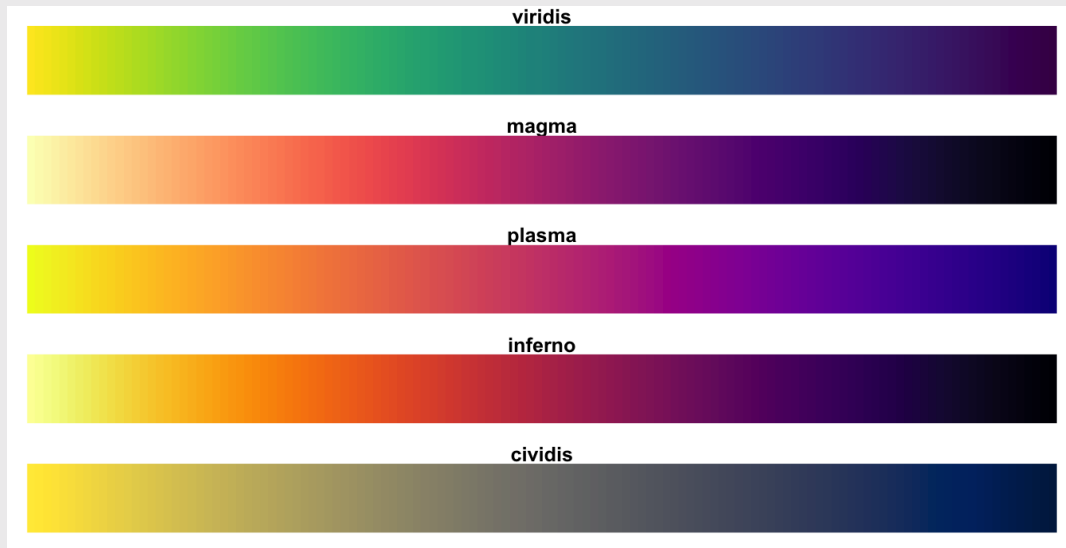
```
library(colorblindr)
cvd_grid(mpg_plot_mycolors)
```



# Use palettes

ColorBrewer

Viridis



## 3 types of palettes

1. Sequential

2. Diverging

3. Categorical



# 3 types of palettes

## 1. **Sequential**

## 2. Diverging

## 3. Categorical

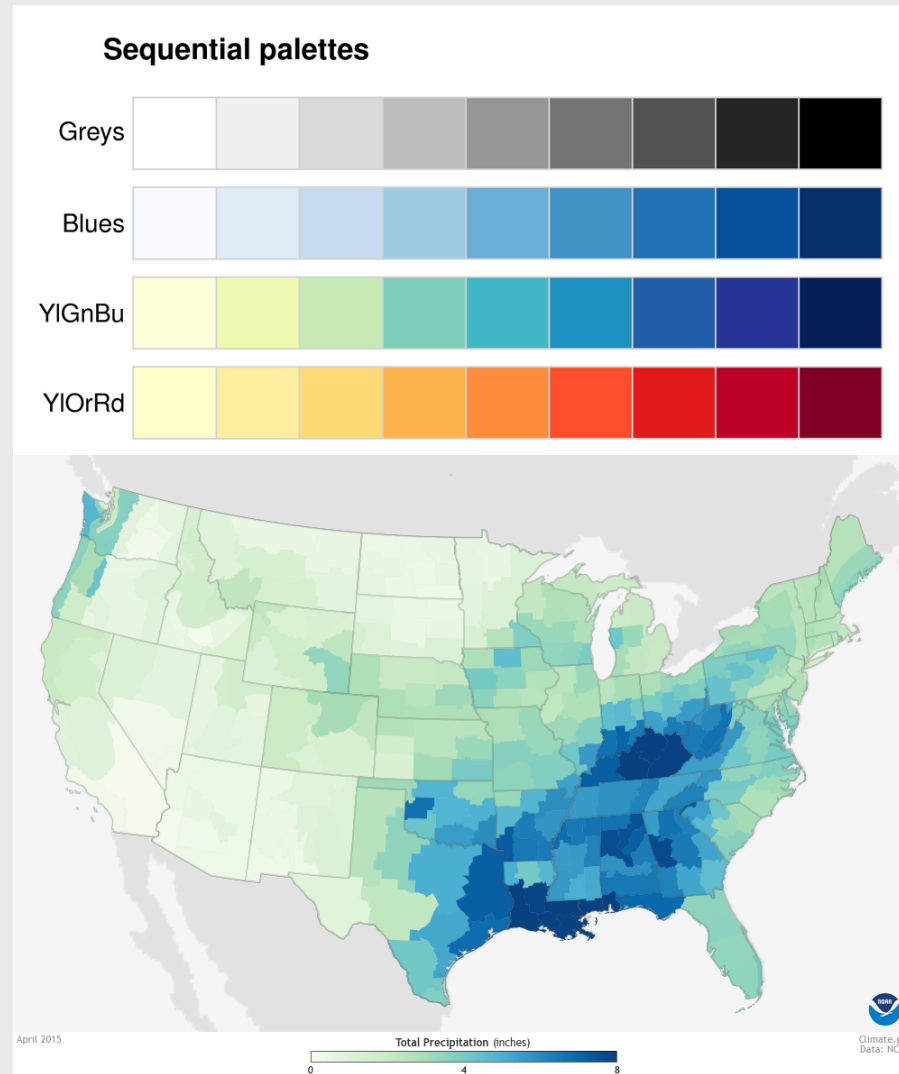


Image from [betterfigures.org](https://betterfigures.org)

# 3 types of palettes

1. Sequential

2. **Diverging**

3. Categorical

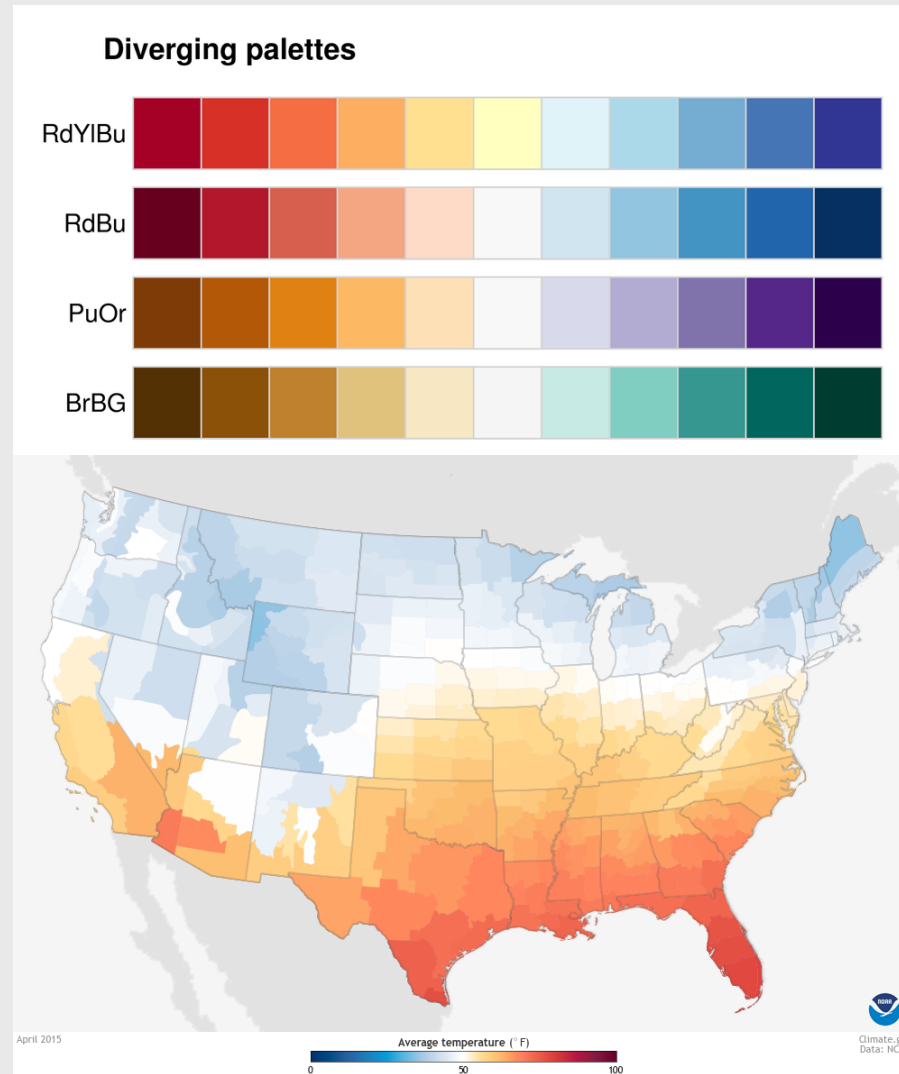


Image from [betterfigures.org](https://betterfigures.org)

# 3 types of palettes

1. Sequential

2. Diverging

3. **Categorical**

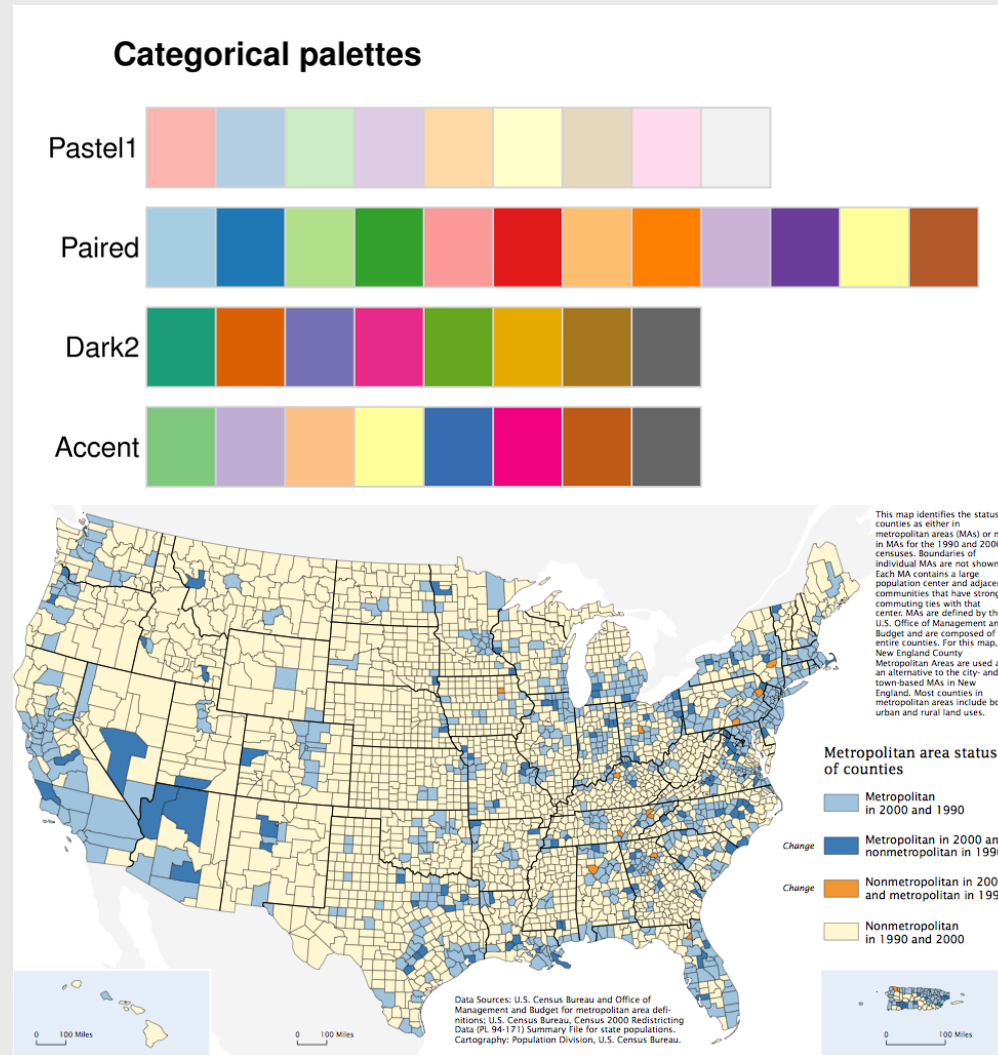


Image from [betterfigures.org](http://betterfigures.org)

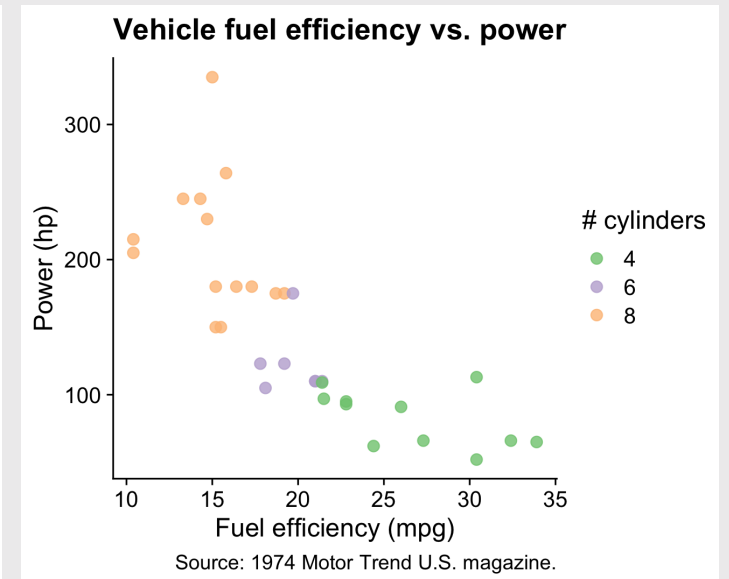
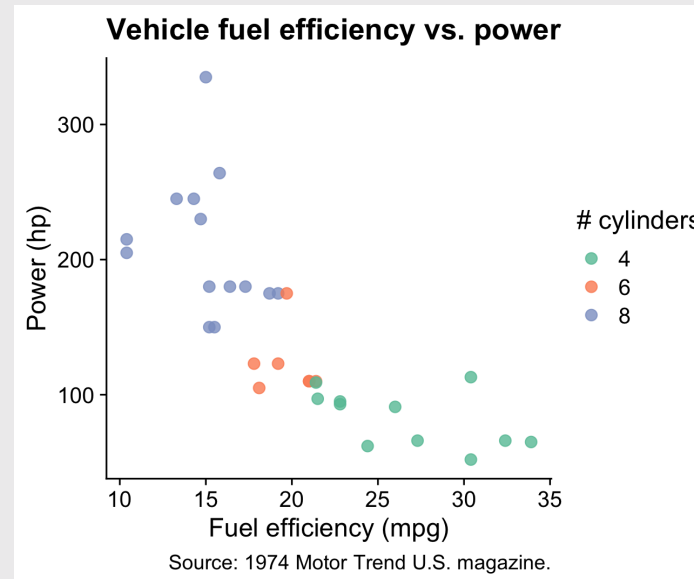
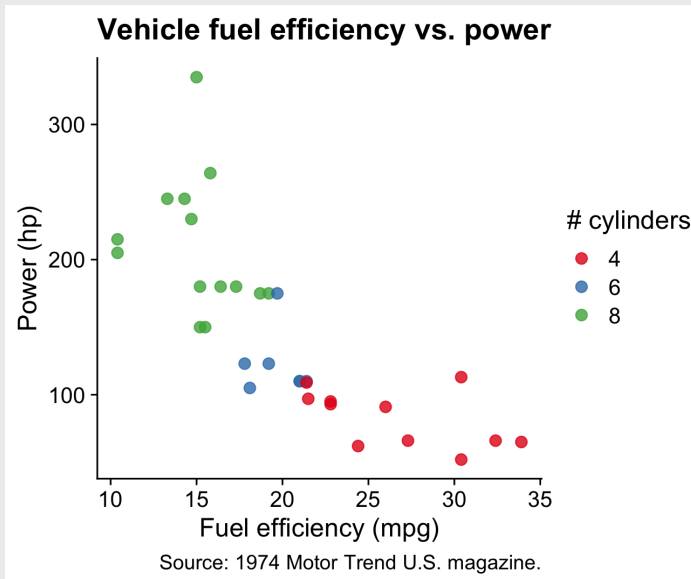


# ColorBrewer palettes

```
mpg_plot +
 scale_color_brewer(
 palette = 'Set1')
```

```
mpg_plot +
 scale_color_brewer(
 palette = 'Set2')
```

```
mpg_plot +
 scale_color_brewer(
 palette = 'Accent')
```



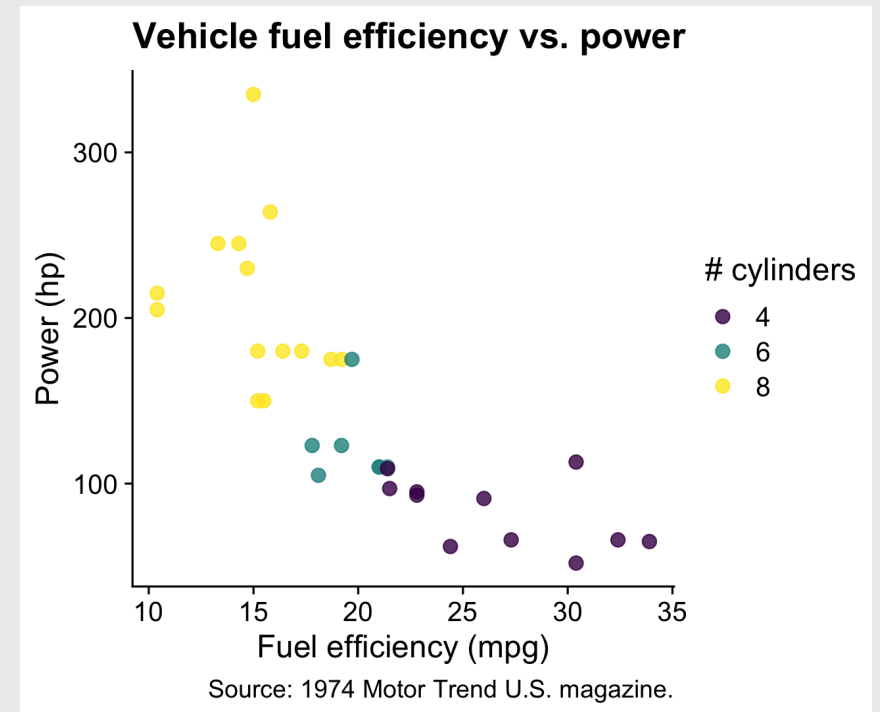
# viridis palettes

Map color to variable

```
mpg_plot <- ggplot(mtcars, aes(x = mpg, y = hp)) +
 geom_point(aes(color = as.factor(cyl)),
 alpha = 0.8, size = 3) +
 theme_half_open(font_size = 16) +
 labs(x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 color = '# cylinders',
 title = "Vehicle fuel efficiency vs. power",
 caption = "Source: 1974 Motor Trend U.S. magazine")
```

Use viridis colors

```
mpg_plot +
 scale_color_viridis(discrete = TRUE)
```

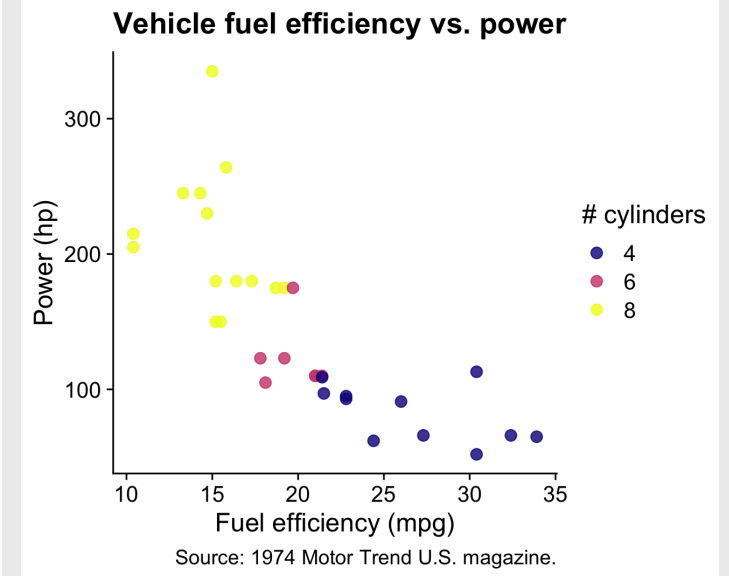
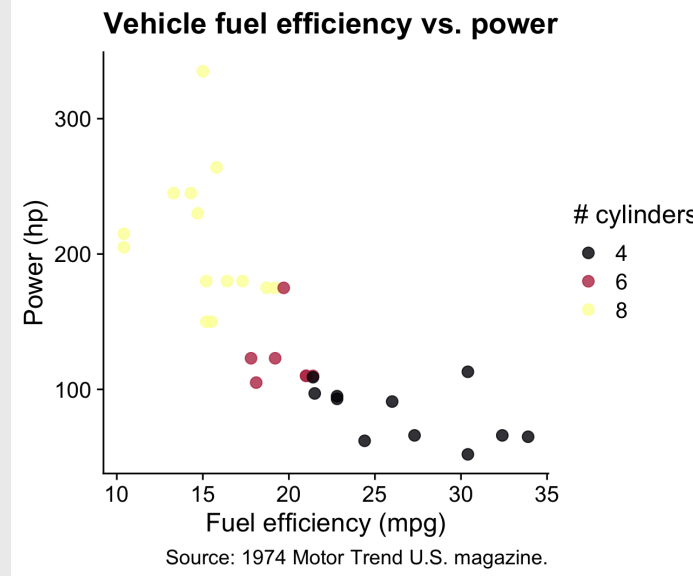
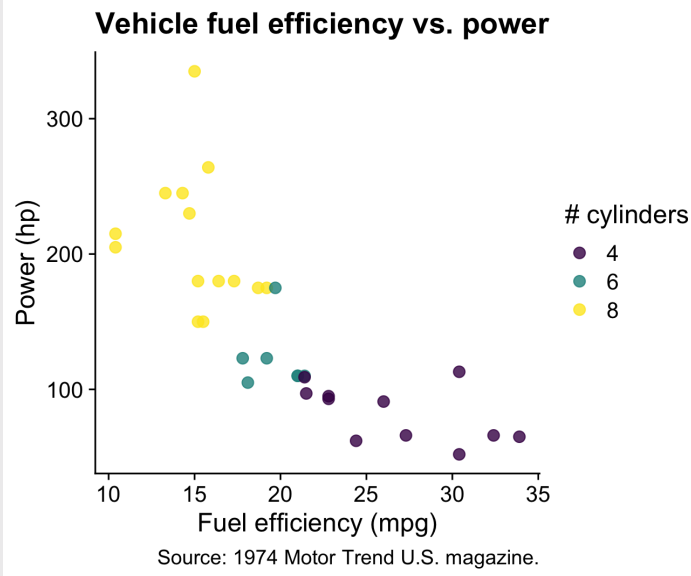


# viridis palettes

```
mpg_plot +
 scale_color_viridis(
 discrete = TRUE,
 option = 'viridis')
```

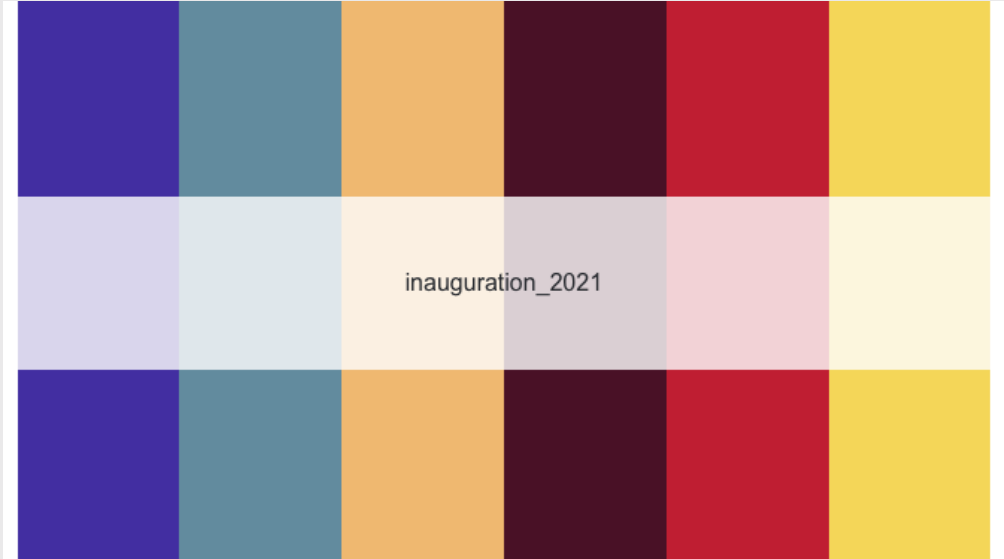
```
mpg_plot +
 scale_color_viridis(
 discrete = TRUE,
 option = 'inferno')
```

```
mpg_plot +
 scale_color_viridis(
 discrete = TRUE,
 option = 'plasma')
```

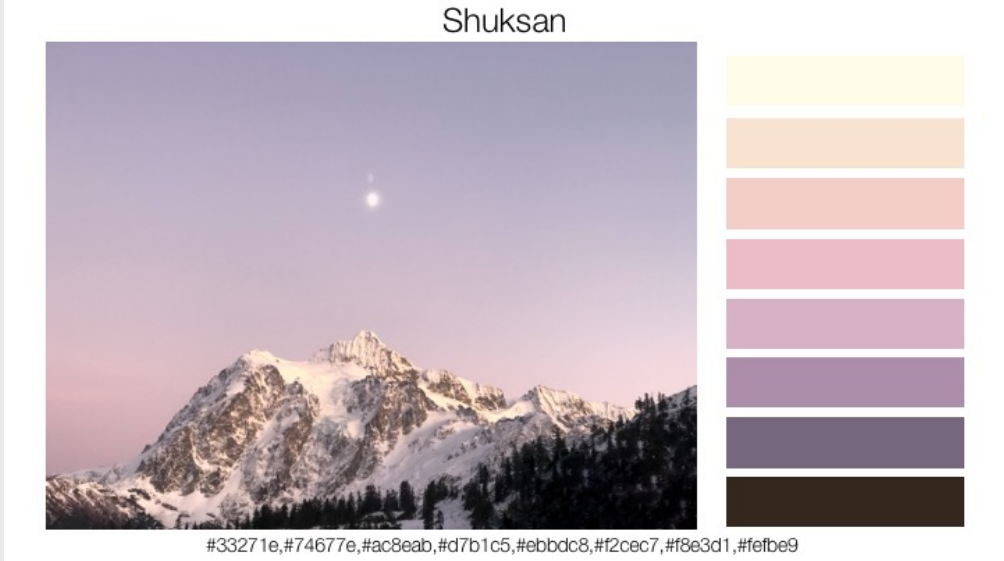


# Fun custom palettes

## Inauguration palette



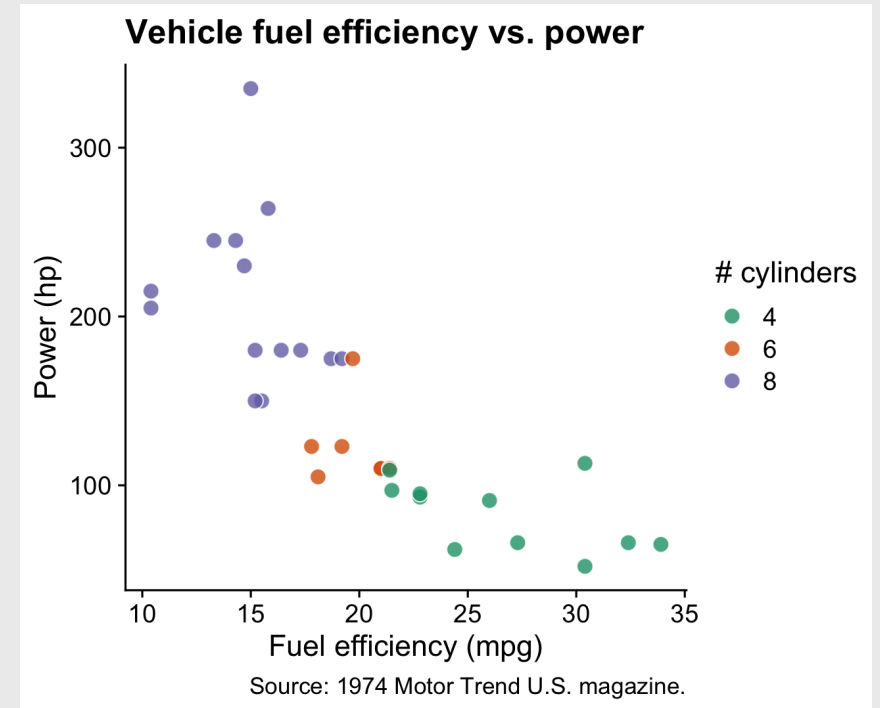
## PNWColors





# Consider using `color` + `fill` for points

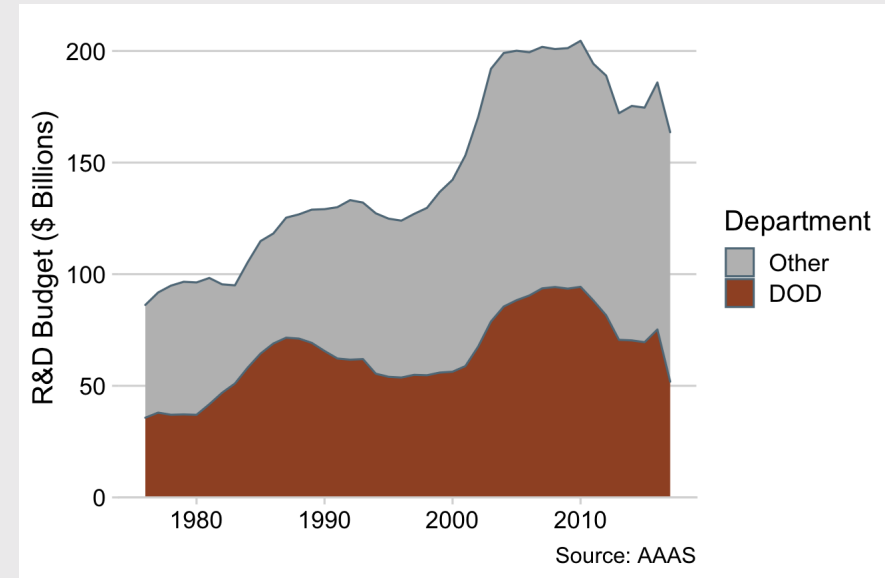
```
ggplot(mtcars, aes(x = mpg, y = hp)) +
 geom_point(
 aes(fill = as.factor(cyl)),
 color = 'white', shape = 21,
 size = 3.5, alpha = 0.8) +
 scale_fill_brewer(palette = 'Dark2') +
 theme_half_open(font_size = 15) +
 labs(
 x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 fill = "# cylinders",
 title = "Vehicle fuel efficiency vs. power",
 caption = "Source: 1974 Motor Trend U.S. magazine.")
```



# grey = "Other"

```
dod_spending_plot <- federal_spending %>%
 mutate(department = fct_other(
 department, keep = 'DOD')) %>%
 group_by(department, year) %>%
 summarise(rd_budget = sum(rd_budget) / 10^3)
 ungroup() %>%
 mutate(department = fct_relevel(
 department, c('Other', 'DOD')) %>%
 ggplot() +
 geom_area(aes(x = year, y = rd_budget,
 fill = department)) +
 scale_y_continuous(
 expand = expand_scale(mult = c(0, 0.05)))
 scale_fill_manual(
 values = c('grey', 'sienna')) +
 theme_minimal_hgrid() +
 labs(x = NULL,
 y = 'R&D Budget ($ Billions)',
 fill = 'Department',
 caption = 'Source: AAAS')
```

dod\_spending\_plot

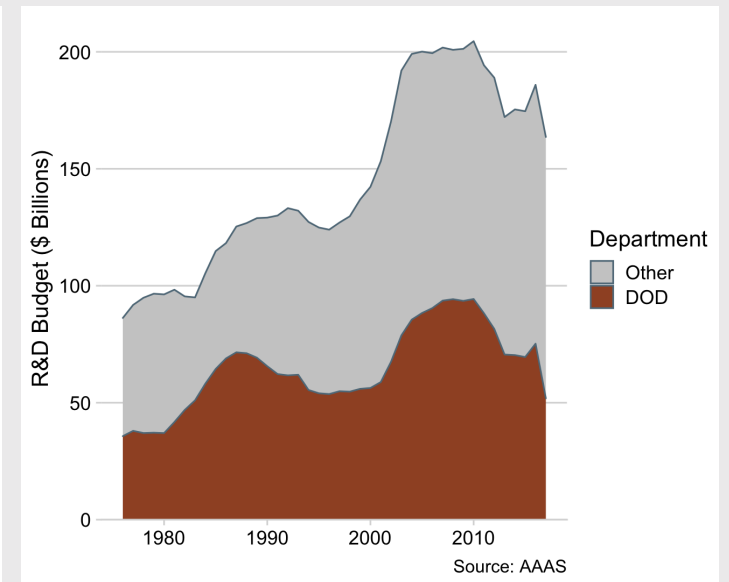
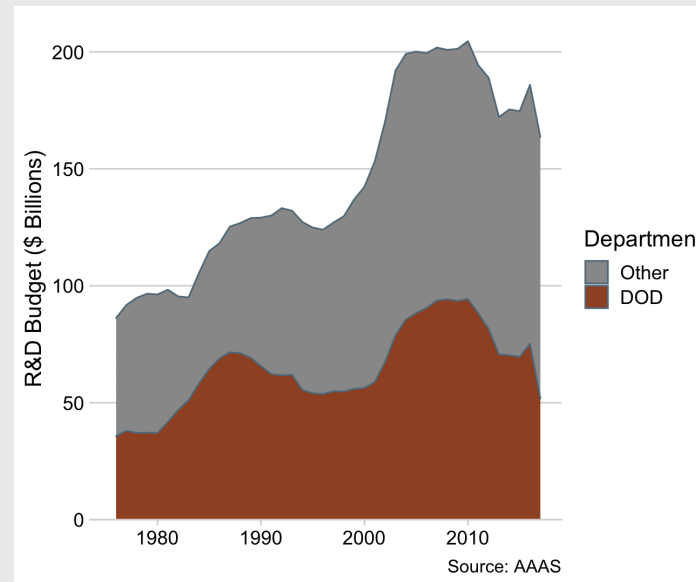
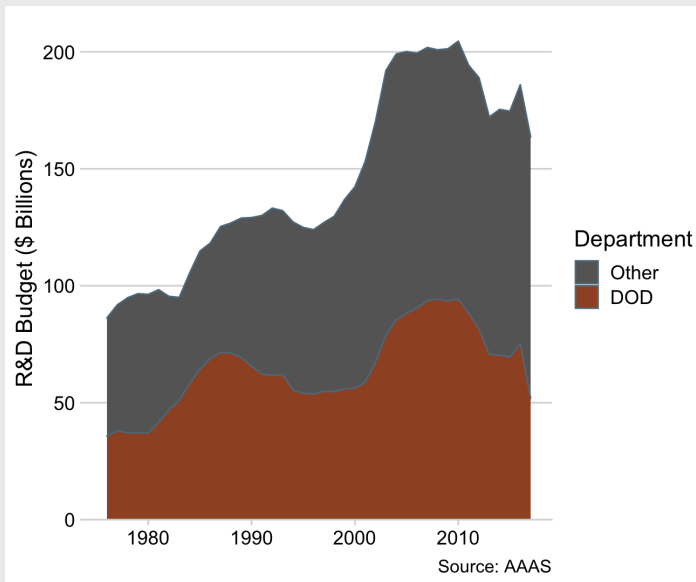


# grey = "Other"

```
dod_spending_plot +
 scale_fill_manual(
 values = c('grey40', 'sienn
```

```
dod_spending_plot +
 scale_fill_manual(
 values = c('grey60', 'sienn
```

```
dod_spending_plot +
 scale_fill_manual(
 values = c('grey80', 'sienn
```

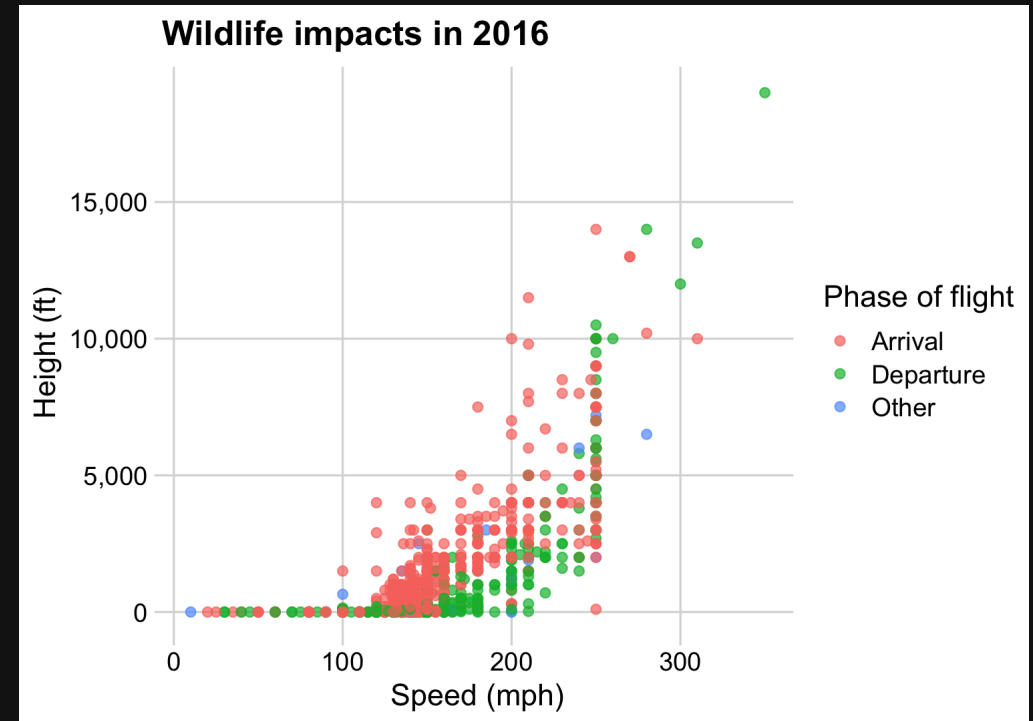


# Your turn

20:00

Make 3 different versions of this chart:

1. Change the colors to the "RdYlBu" ColorBrewer palette.
2. Change the colors to the "inferno" palette from the **viridis** library.
3. Change the colors to your custom triadic palette:
  - Use the ["eye dropper"](#) tool in Google Chrome to select a color from a website, then
  - Use your color and the [color wheel tool](#) to find a triadic color palette.



# Week 10: *Polishing Charts*

1. Scales

2. Annotations

BREAK

3. Colors

4. **Fonts**

5. Rmd tricks

# Fonts matter



"Fast Taco"



"Mega Flicks"

**FONTS MATTER**

You'll Always  
Be Mine

YOU'LL  
ALWAYS  
BE MINE

Best resource on fonts:  
[practicaltypography.com](https://practicaltypography.com)



# Font families you should consider using

Roboto

Download:

Source

- Individually from <https://fonts.google.com/>
- All of these with [this zip file](#)

Fira

Alegreya

Lato

# Use fonts to create **hierarchy**

```
Hierarchy
```

```
Hierarchy
```

```
Hierarchy
```

```
Hierarchy
```

Hierarchy

Hierarchy

Hierarchy

**Hierarchy**

## Title

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

# Size

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## **Weight**

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## Color

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## Spacing

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

## Typeface

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words:

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.



# Title

## Subtitle

This is some text that goes into detail and explains a lot more about the topic described in the title. Here's some random Latin words: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

# Use fonts with **same-height** numbers

"Oldstyle" (bad)



1 2 3 4 5 6 7 8 9 0

The image shows the numbers 1 through 0 in an oldstyle font. The numbers 4, 5, 6, 7, 8, and 9 are significantly shorter than the numbers 1, 2, 3, and 0. Red boxes are drawn around the top and bottom of the shorter numbers to illustrate their inconsistent height relative to the taller numbers.

Lining (good)



1 2 3 4 5 6 7 8 9 0

The image shows the numbers 1 through 0 in a lining font. All numbers have the same height. A red box is drawn around the number 4 to illustrate its consistent height with the other numbers.

# Use fonts with **same-width** numbers

"Proportional" (bad)



1|2|3|4|5|6|7|8|9|0|

"Tabular" (good)



1|2|3|4|5|6|7|8|9|0|

# How to customize fonts in **ggplot**

# 1. Change the whole theme

For "Base R" themes, use `base_family`:

```
theme_minimal(base_family = "Roboto Condensed")
```

```
theme_bw(base_family = "Roboto Condensed")
```

For "cowplot" themes, use `font_family`:

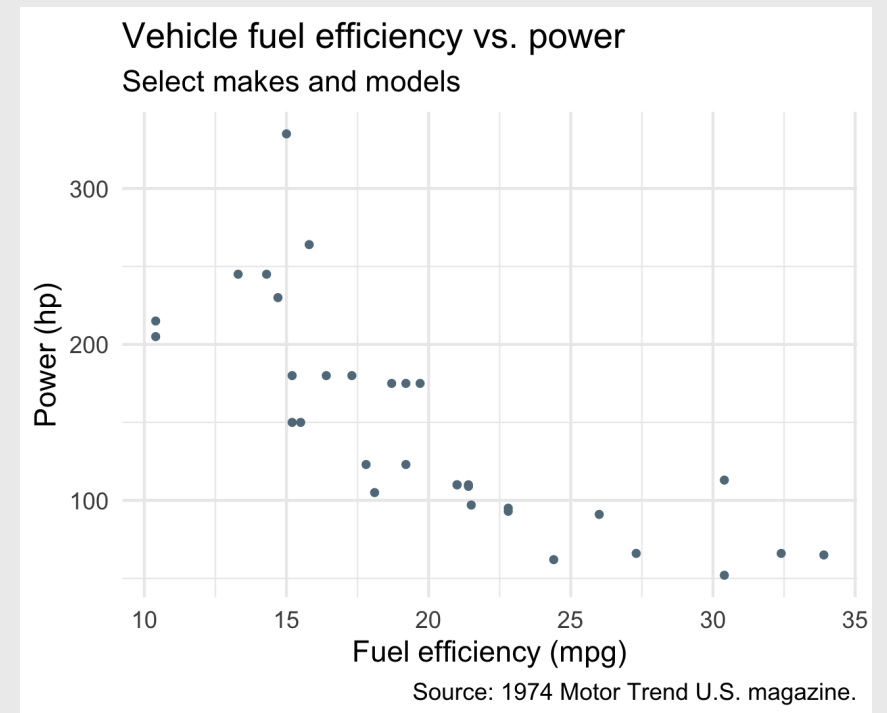
```
theme_half_open(font_family = "Roboto Condensed")
```

```
theme_minimal_grid(font_family = "Roboto Condensed")
```

# 1. Change the whole theme font

Make the base plot

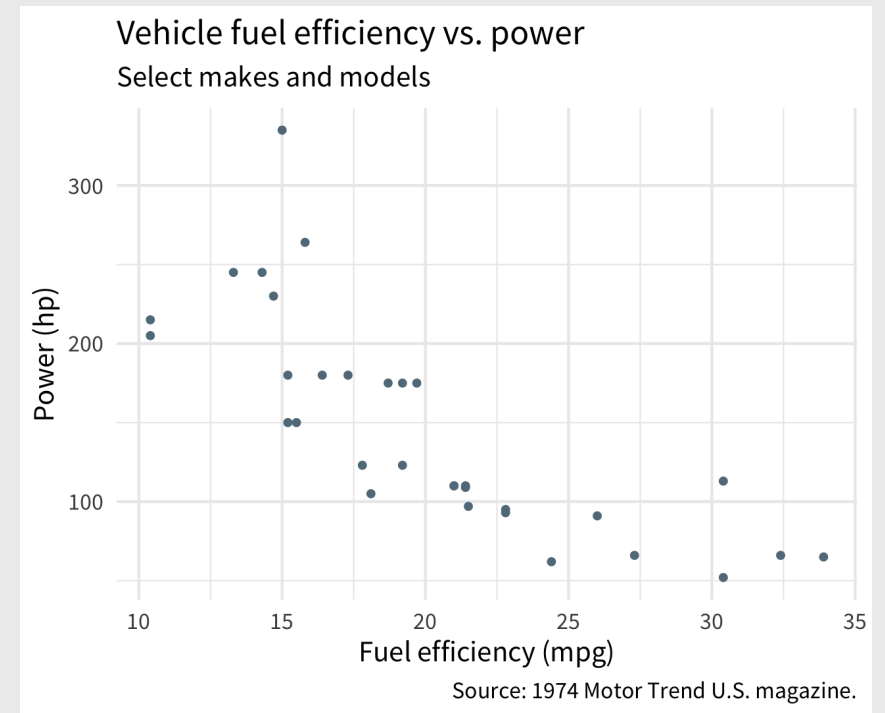
```
mpg_plot <- ggplot(mtcars) +
 geom_point(aes(x = mpg, y = hp)) +
 theme_minimal(base_size = 15) +
 labs(
 x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 title = "Vehicle fuel efficiency vs. power",
 subtitle = "Select makes and models",
 caption = "Source: 1974 Motor Trend U.S. magazine.")
```



# 1. Change the whole theme font

Use `base_family` with base themes

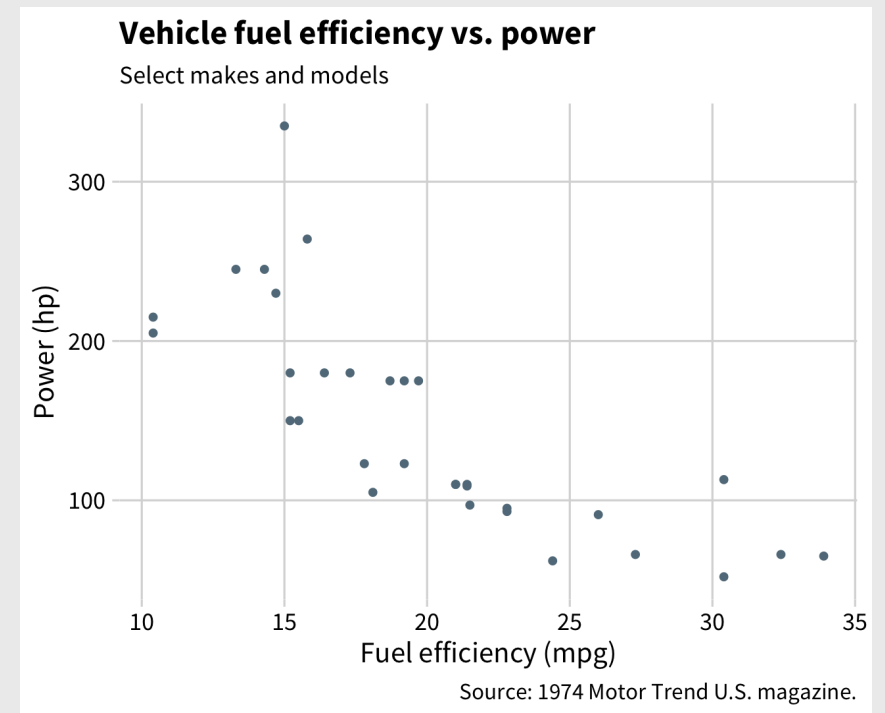
```
mpg_plot <- ggplot(mtcars) +
 geom_point(aes(x = mpg, y = hp)) +
 theme_minimal(
 base_family = 'Source Sans Pro',
 base_size = 15) +
 labs(
 x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 title = "Vehicle fuel efficiency vs. power",
 subtitle = "Select makes and models",
 caption = "Source: 1974 Motor Trend U.S. magazine.")
```



# 1. Change the whole theme font

Use `font_family` with cowplot themes

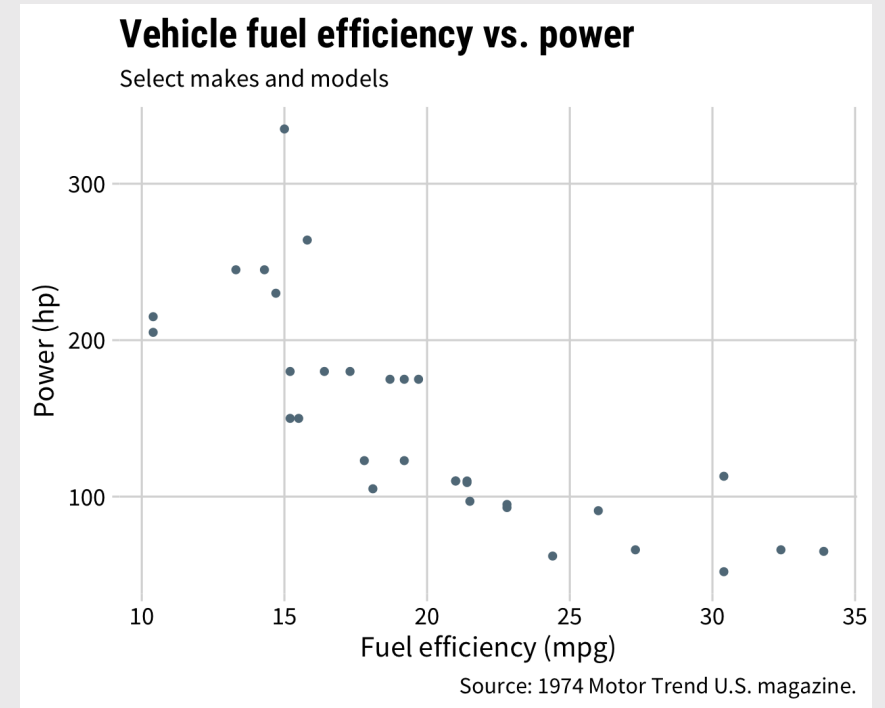
```
mpg_plot <- ggplot(mtcars) +
 geom_point(aes(x = mpg, y = hp)) +
 theme_minimal_grid(
 font_family = 'Source Sans Pro',
 font_size = 15) +
 labs(
 x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 title = "Vehicle fuel efficiency vs. power",
 subtitle = "Select makes and models",
 caption = "Source: 1974 Motor Trend U.S. magazine.")
```





## 2. Adjust theme elements with `element_text()`

```
mpg_plot <- ggplot(mtcars) +
 geom_point(aes(x = mpg, y = hp)) +
 theme_minimal_grid(
 font_family = 'Source Sans Pro',
 font_size = 15) +
 theme(
 plot.title = element_text(
 family = "Roboto Condensed",
 size = 20)) +
 labs(
 x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 title = "Vehicle fuel efficiency vs. power",
 subtitle = "Select makes and models",
 caption = "Source: 1974 Motor Trend U.S. magazine.")
```



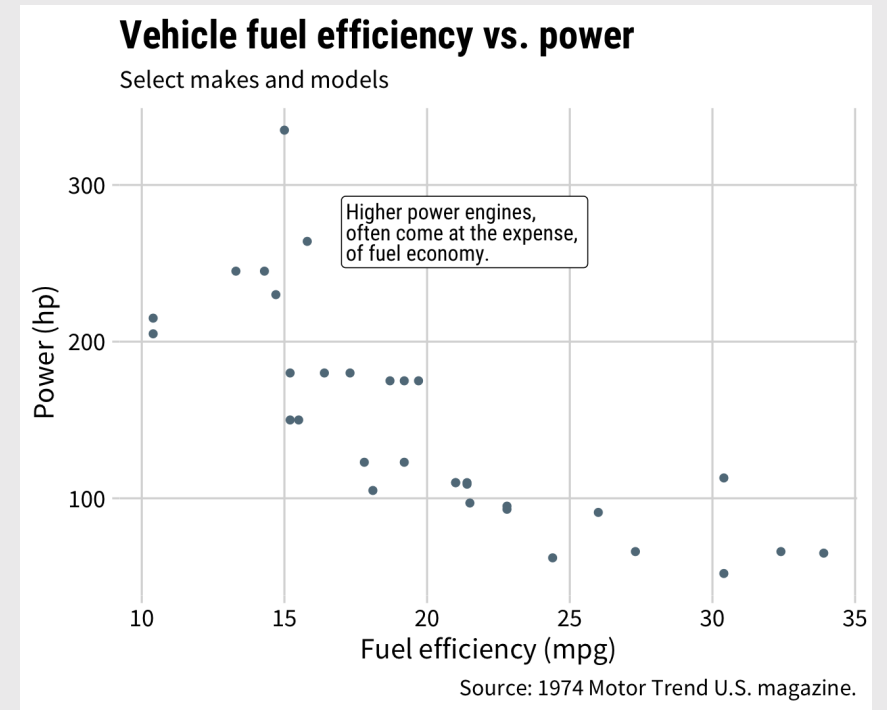
See theme components [here](#)

### 3. Adjust annotations:

`geom_text()`, `geom_label()`, and `annotate()`

```
label <- "Higher power engines,
often come at the expense,
of fuel economy."

mpg_plot +
 geom_label(
 data = data.frame(
 x = 17, y = 270, label = label),
 aes(x = x, y = y, label = label),
 lineheight = .8, hjust = 0,
 family = 'Roboto Condensed')
```



# The `hrbrthemes` package:

Great themes + great fonts

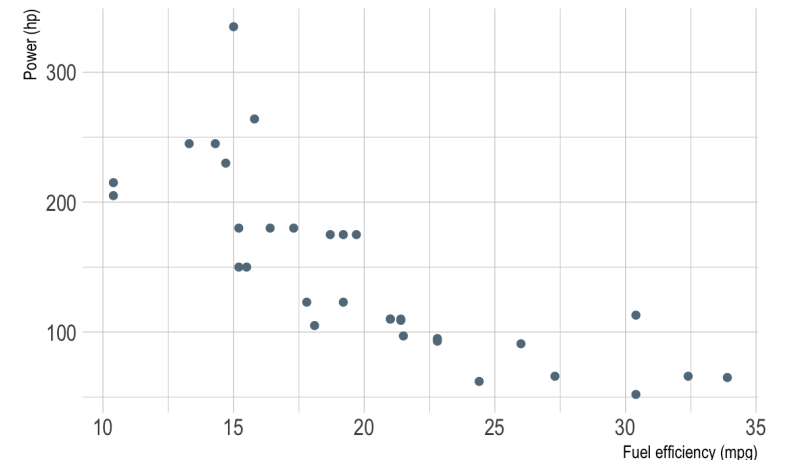
```
library(hrbrthemes)
```

```
mpg_plot <- ggplot(mtcars) +
 geom_point(aes(x = mpg, y = hp)) +
 labs(
 x = "Fuel efficiency (mpg)",
 y = "Power (hp)",
 title = "Vehicle fuel efficiency vs. power",
 subtitle = "Select makes and models",
 caption = "Source: 1974 Motor Trend U.S. magazine"
```

```
mpg_plot +
 theme_ipsum()
```

## Vehicle fuel efficiency vs. power

Select makes and models



Source: 1974 Motor Trend U.S. magazine.

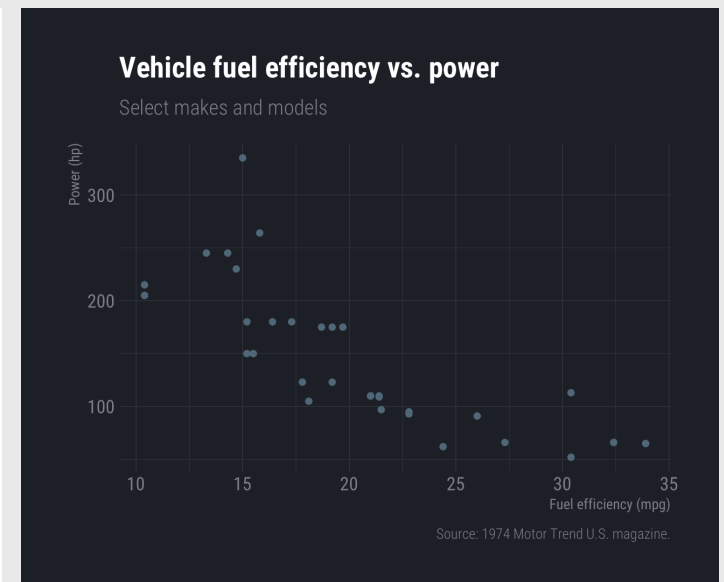
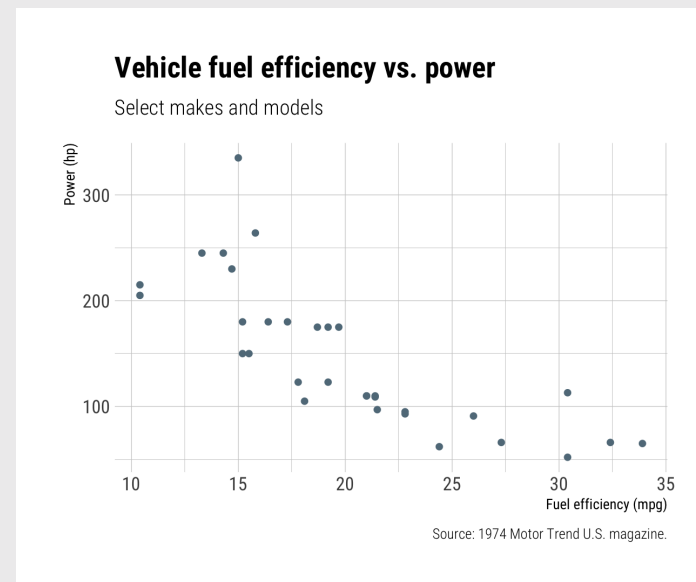
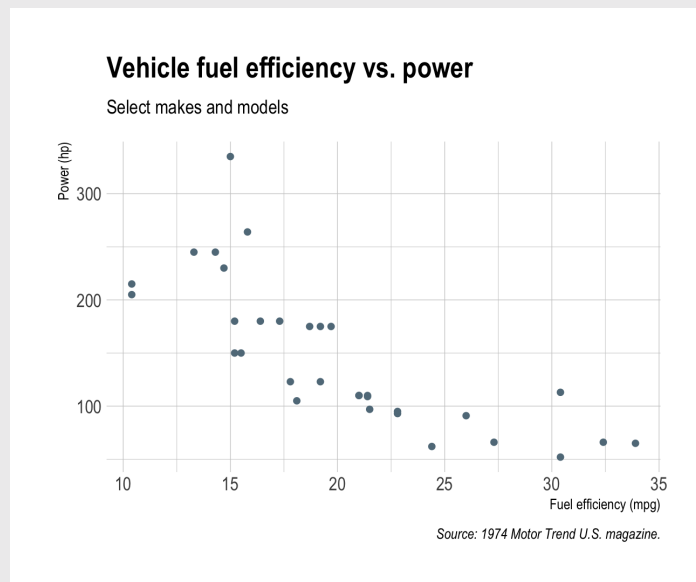
# The `hrbrthemes` package:

## Great themes + great fonts

```
mpg_plot +
 theme_ipsum()
```

```
mpg_plot +
 theme_ipsum_rc()
```

```
mpg_plot +
 theme_ft_rc()
```



# Your turn

15:00

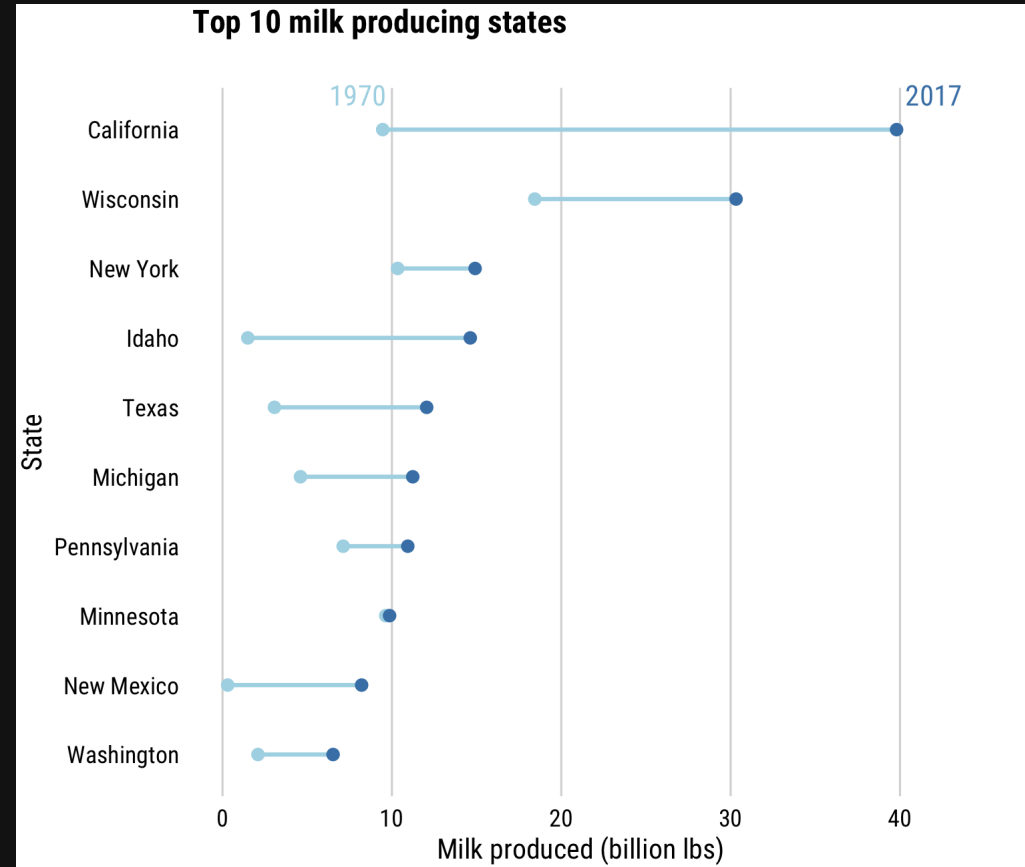
Modify the fonts and annotations in the dumbbell chart to match the chart shown here. The main font is 'Roboto Condensed'.

Once you've recreated the plot, try other fonts and themes, such as:

- The 'Source Sans Pro' font.
- The 'Lato' font.
- The `theme_ipsum()` theme from the `hrbrthemes` library.

Hint: Use `annotate()` to insert the year labels at the top:

- 1970: `x = 9, y = 10.5`
- 2017: `x = 40, y = 10.5`



# Week 10: *Polishing Charts*

1. Scales

2. Annotations

BREAK

3. Colors

4. Fonts

5. Rmd tricks

# Use themes to change global "look"

## Default "bootstrap" themes

Change theme in YAML header:

```
title: "your title"
output:
 html_document:
 theme: united
 highlight: tango
```

[Preview themes here](#)

## rmdformats package

```
install.packages("rmdformats")
```

```
title: "your title"
output: rmdformats::<template name>
```

# Set a global chart theme

```
theme_set(theme_minimal_grid())
```

Check out [Pimp my RMD](#), by Yan Holtz